Making Software, Making Regions: Labor Market Dualization, Segmentation, and Feminization in Austin, Portland and Seattle

Dillon Mahmoudi
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds

Part of the Computer Engineering Commons, and the Human Geography Commons

Let us know how access to this document benefits you.

Recommended Citation
https://doi.org/10.15760/etd.5652

This Dissertation is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.
Making Software, Making Regions: Labor Market Dualization, Segmentation, and Feminization in Austin, Portland and Seattle

by

Dillon Mahmoudi

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Urban Studies

Dissertation Committee:
Greg Schrock, Chair
Sy Adler
Amy Lubitow
Nathan McClintock

Portland State University
2017
© 2017 by Dillon Mahmoudi

Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

http://creativecommons.org/licenses/by-nc-nd/4.0/
Abstract

Through mixed-methods research, this dissertation details the regionally variegated and place-specific software production processes in three second-tier US software regions. I focus on the relationship between different industrial, firm, and worker production configurations and broad-based economic development, prosperity, and inequality. I develop four main empirical findings.

First, I argue for a periodization of software production that tracks with changes in software laboring activity, software technologies, and wage-employment relationships. Through a GIS-based method, I use the IPUMS-USA to extensively measure the amount and type of software labor in industries across the US between 1970 and 2015. I map the uneven geography of software labor that produces different clusters of various software occupations. Second, I argue that between each software period, locational windows provide an opportunity for second-tier software regions to challenge Silicon Valley. I combine the IPUMS-USA dataset with interviews of software workers to analyze forms of regionally specific modes of production in Seattle, Washington, Portland, Oregon, and Austin, Texas. I trace how software production in these three cities evolves between each software period, taking on different spatial configurations, firm strategies, labor practices, and technological characteristics. Third, I argue that software labor is hyper-sensitive to deskilling because of software production activity produces software. I combine occupation classifications and interviews with software workers to interrogate the ever-present need for software workers to learn the newest development practices and software
languages as firms seek to automate software production. I define five key moments since the 1970s that exemplify software labor market dualization and segmentation.

Using interviews, and conference observations, I find that community-based organizations and labor market intermediaries locally mitigate the structural tendencies toward labor market dualization and segmentation. I argue that without intervention, the layered and bifurcated labor market for software production reproduces existing inequalities. Further, the organizational, technological, and spatial changes in software production reduce the potential for equitable wealth production. Ultimately, this dissertation argues for the importance of labor organizing in software, contributing empirical and theoretical work in a lineage of regional-based industrial restructuring literature. The regional and industrial geographies produced by and out of software production are significant forces in the economy at regional and national scales. I connect this process to the feminization of other industries, noting how the technical nature of software production structurally genders and racializes the labor force. Leveraging a labor feminization framework highlights the flexibilization of labor and the rift between the pace of software skill building and technological development.

Both software production and regional economies are necessary entry points to understand new capitalist relations. Understanding these new relations thus requires examining how configurations of software production differ across regions, how they impact industry and regional economic development outcomes, and how they weaken or strengthen actions of local workers, local organizations, and local firms. These processes offer a glimpse into how the contemporary moment of production differs from other moments of production. Armed with this understanding, this research will be able to connect industry and regional economic-development outcomes to regionally specific modes of production, answering relevant software-based economic-development policy questions.
Dedication

For my family.
Acknowledgements

Knowledge production never happens in isolation and this dissertation represents a journey fueled by inspirational people. Throughout this process, I have tremendously benefited from the selfless generosity, guidance, and support of my interview subjects, my committee, my mentors, my colleagues, and my family.

I am appreciative of, and owe a great debt to, the interview subjects who volunteered time away from work and loved ones. Most interviews were about an hour, but 14 spiraled into 2+ hour long conversations filled with stores of the pressures to know new software, sexism in the workplace, and other personal stories of career growth and career trajectories.

I am also indebted to the people that helped me complete my dissertation and all the necessary steps along the way. I am not only thankful that each member of my dissertation committee agreed to participate as committee member, but also thankful for their immense generosity, patience, and trust. I am particularly indebted to my chair, Greg Schrock, for his unwaivering support, guidance, mentoring, friendship, careful teaching, and for lengthy technical discussions of the rarely used PUMA datasets. He has always made himself available to me, even prior to joining the Toulan School. Nathan McClintock was instrumental in challenging me to engage more meaningfully with the theories I sought to use, doing so with tact, patience, and a smile. Similarly, Amy Lubitow provided incisive feedback, pushing me to broaden disciplinary boundaries and improve my methodological rigor, while artfully providing papers along the way so that I wouldn't...
get lost. With Greg, Nathan, and Amy, I will miss the conversations while making half-caff coffee, having flat whites, sipping a pint, eating burritos, scarfing pizza in public libraries, debating north arrows, and most importantly, conducting research where they selflessly chose to include me. Lastly, Sy Adler, the committee member with whom I have known the longest, has given me years of mentoring and instruction, always challenging my writing and asking for me to simultaneously think bigger and be more precise in my language. As a testament to Sy’s instruction and mentoring, I voluntarily enrolled in his Advanced Planning Theory class multiple times, formally the first time then as reading and conference. Together, they have been a model committee. Through this process, they’ve taught me how I might interact with my future students: through encouragement, rigor, humor, continued probing, and endless inspiration.

I owe a great deal to the other mentors I’ve had along the way. Lisa Bates always quick to provide feedback via red pen and a cackle. Joe Cortright for the continued mentoring and from whom I’ve learned so much about both datasets and the Portland Timbers. Gerry Sussman and Ethan Seltzer for agreeing to write papers with me. Paula Carder for the skeptical encouragement. Jennifer Dill for first agreeing to hire me to do research at Portland State to whom I owe learning what a proper literature review was. Jason Jurjevich for the opportunity to TA a class and with whom I look forward to collaborating with in Baltimore. Thad Miller for including me on the Futurescapes project. Numerous other faculty and staff in Urban Studies and Geography have given me great support: Carl Abbott, James Strathman, Hunter Shobe, David Banis, Charles Heying, Heejun Chang, Andree Tremoulet, Kevin Martin, and Ryan Gratzer. Faculty outside of PSU have also provided their time, support, and encouragement in selfless fashion that I will certainly invoke when a graduate student sends me an email or talks to me at a conference: Brett Christophers, Elvin Wyly, Jamie Peck, Betsy Donald, Jim Thatcher, David O’Sullivan, Richard Walker, Kate Nesse, Matt Wilson, Katherine Rankin, Mike Powe, Emily Talen,
Josh Akers, Marc Doussard, Jason Hackworth, Chris Muellerleile, and Faranak Miraftab.

I have learned so much from fellow students. The dissertation zone group – Ellie Harmon, Erin Goodling, and Anthony Levenda – was an endless source of support. I was the laggard of the group, but we all now have PhDs. Jamaal Green has been a source of constant surly questioning, always making me rethink my position, and always available for a beer. RJ Koscielniak sees through the nonsense and flips tables. Moriah McSharry McGrath welcomed me into the program at PSU and helped me figure out what graduate school was. Amy Coplen made sure I didn’t forget it, and never let me be lazy about scholarship or our own graduate student labor power. Marissa Matsler, Diana Denham, Steven Howland, Arlie Adkins, Jacklyn Kohon, Erin Looper, and Dan Larson have each forced me to be a better scholar. I am indebted to the students with whom I’ve met at conferences, many of whom who are now faculty across the world and have introduced me to new literatures and provided crucial feedback at conferences. Emily Rosenman provided last minute dissertation feedback and advice as we both struggled to complete dissertations on the same timeline. My geofam – Patrick Bigger, Sophie Webber, Heather Whiteside, Kelly Kay, Taylor Shelton, Adam Romero, and Luis Felipe Alvarez León – have included me as one of their own. Allison Forbes gave me pointed dissertation feedback, steering my dissertation in new directions. Ingrid Behrsin, Alex Tarr, Sarah Knuth, Will Payne, Dominic Prestifilippo, and TJ Lowdermilk all have kindly told me when I need to rethink something. There are many more names at PSU and abroad, including those at the Summer Institute of Economic Geography, the Revisiting Critical GIS meeting, the Dimensions of Political Ecology conference, the Oxford Internet Institute Summer School, Hackademia, the Critical Geographies mini-conference, the Radical Planning mini-conference, the Left Coast Political Ecology group, and the Resistance GIS conference.

Lastly, my family has so been a source of much love and encouragement, with whom
none of the other acknowledgments would even be remotely possible. Late nights with gummy bears, the late nights with Aurora, and so much patience. This is for them.
Table of Contents

Abstract i
Dedication iii
Acknowledgements iv
List of Tables x
List of Figures xii

1 Introduction 1

2 Literature Review 8
   2.1 Framework for Regional Analysis of Software Production ............. 8
   2.2 Labor Markets and the Labor Process ................................ 12
   2.3 Regional Development and Industrial Restructuring ...................... 22
   2.4 Existing Research on Regional High-Tech Production .................... 30
   2.5 Emerging Research Questions ............................................. 36

3 Research Design and Methodology 45
   3.1 Introduction ................................................................. 45
   3.2 Research Design ............................................................. 47
   3.3 Research Questions and Method .......................................... 49
   3.4 Case Selection ............................................................... 58
   3.5 Data Collection and Data Sources ...................................... 63

4 The Restructuring of Software Production 71
   4.1 Introduction ................................................................. 71
   4.2 Software Periods and Software Production ................................ 73
   4.3 1950s and 1960s: Contracting, Increased Competition, and Punch Cards 75
   4.4 1970s and 1980s: Corporate Software, Memory and Databases .......... 88
   4.5 1990s and 2000s: Mass-market Software, Desktop Computers, AOL .... 98
   4.6 Late 2000s: Cloud, Mobile, Big Data .................................... 110
   4.7 Conclusion .................................................................. 117
List of Tables

2.1 Regional characteristics of software industrial districts. .......................... 41
2.2 Firm and production characteristics of software industrial districts. ............. 42
2.3 Labor characteristics of software industrial districts. .............................. 43

3.1 Synthesizing mixed-methods research. .................................................. 59

4.1 Periods of software. ................................................................. 76
4.2 Capitalist variants. ........................................................................... 77
4.3 Software programming language generations. ........................................... 86
4.6 Measuring concentration of software workers by industry, 1970–2015. ........ 107
4.7 Diffusion of software labor across industries, 1970–2015. ......................... 108

5.1 Metro software agglomerations by year, number of software workers, 1970–2015. 127
5.2 Metro software agglomerations by year, share of labor force, , 1970–2015. .... 128
5.3 Technological characteristics of software industrial districts. ....................... 134
5.5 Industrial composition of software workers in the Portland region, ACS 2011–2015. .......... 147
5.6 Industrial composition of software workers in the Austin region, ACS 2011–2015. .......... 160
5.7 Comparing industrial districts of Austin, Portland, and Seattle. .................... 168
5.9 Software occupation classification, ACS 2011–2015. ............................... 173

6.1 Bifurcation of software labor, 1970 to 2015. ........................................... 194
6.4 Median income by gender and race/ethnicity for software workers, Austin, Portland, Seattle, ACS 2011–2015 ......................................................... 229
6.5 Software occupation labor market tiers, ACS 2011–2015. ......................... 230
6.6 Median income labor market tiers by gender for software workers, Austin, Portland, Seattle, ACS 2011–2015 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 231
6.7 Median income labor market tiers by race/ethnicity for software workers, Austin, Portland, Seattle, ACS 2011–2015 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232
List of Figures

2.1 Idealized Marshallian industrial district. ................................. 39
2.2 Idealized hub-and-spoke industrial district. .......................... 39
2.3 Idealized hub-and-spoke industrial district. .......................... 40
2.4 Idealized public-sector industrial district. ............................. 40

4.1 A framework for connecting capitalist variants, product cycles, and industrial districts across four periods of software. ......................... 78
4.2 IBM’s market share and projected growth. Fortune Magazine, 1968. .... 90
4.3 Diffusion of software workers across selected industries, 1970–2015. 103
4.3 Dotcom boom and bust, NASDAQ Composite INDEX (IXIC) 1997 to 2017. 106

5.1 Stylized portrayal of Seattle’s software industrial district. ............ 138
5.2 Stylized portrayal of Portland’s software industrial district. .......... 148
5.3 MentorGraphics job wanted ad in The Oregonian, 1982. .............. 150
5.4 Stylized portrayal of Austin’s software industrial district. .......... 161
5.5 Income density curves for software workers, Austin, Portland, and Seattle. 171
5.6 Software occupation share of region labor force, stacked, ACS 2011–2015. 177

6.1 Significant occupation recoding of software labor by occupation code, 1970–2015. .......................................................... 193
6.2 National income distribution by gender for software workers, ACS 2011–2015. 200
6.3 National income distribution by race/ethnicity for software workers, ACS 2011–2015. .......................................................... 201
6.5 Income distribution by race/ethnicity for software workers, Austin, Portland, and Seattle, ACS 2011–2015. .............................................. 205
6.6 Income distribution by labor market tier by gender for software workers, Austin, Portland, and Seattle, ACS 2011–2015. ...................... 208
6.7 Income distribution by labor market tier by race/ethnicity for software workers, Austin, Portland, and Seattle, ACS 2011–2015. ...................... 209
6.8 Count of software workers in each labor market tier, Austin, Portland, and Seattle, ACS 2011–2015. .................................................... 233
1 Introduction

Learning these skills [software programming] isn’t just important for your future. It’s important for our country’s future. If we want America to stay on the cutting edge, we need young Americans like you to master the tools and technology that will change the way we do just about everything.

President Barack Obama on YouTube for Code.org 2014

Software is everywhere and seemingly in everything. Software is part of every tech gadget that we use both inside and outside of work. Software is even core to new products and core to the evolution of established products: even our cars are now receiving regular software updates. The growth of software from obscure military and business venture to near ubiquity in daily life means that software production is a fundamental part of the economy. Firms that employ large numbers of software workers like Google, Microsoft, Apple, and Amazon dominate the news headlines. Investors cling to reports of their earnings and make predictions on the global economy based on the health of these “tech” companies even though, by industrial classification, they are classified as a search engine firm, a software publisher, a computer manufacturer, and a wholesale distributor.

To keep pace with the demand of software workers, a new set of trade schools has promulgated “code schools” into our vocabulary and even liberal arts colleges provide computer science degrees. Veteran workers and new workers clamor to learn software development skills and new software languages in a frantic frenzy to both develop software and stay relevant in the ever-changing software landscape. Even politicians have jumped
on the software development bandwagon. During a computer programming media event in December 2014, President Barack Obama became the first US President to “code”. Afterward, the White House released a YouTube video in which President Obama addressed Americans highlighting the importance of learning software programming (chapter epigraph).

President Obama’s focus on software programming skills and the competitive nature of the US economy underscores the rapid growth of software production as a whole. Since software was first produced in the private sector in the 1950s, the number of laborers that produce software, the number firms that develop software for in-house purposes, and the number of firms that produce software has grown faster than analysts’ expectations. Future predictions anticipate firms, governments, and individuals to spend $2.4 trillion on technology in 2017, including about $1 trillion in North America alone. These tech expenditure figures include hardware, like computers and printers, alongside software. However, by 2020 analysts expect the software portion of technology expenditures to surpass the hardware portion, accounting for more than $1.2 trillion on software alone (Sharwood 2017).

Thus, with a focus on programming skills, workers are the intended audience of President Obama’s message. The message is intended to serve two related purposes. First, to ensure American workers have the skills necessary so that American firms are the beneficiary of increased software expenditures. Second, to provide a source of jobs for American workers. Yet, the focus on software skills treats the software production as a simple supply and demand function. The relationship between labor markets, firms, technology, and the location of software production is more complicated than solving an equilibrium equation between labor supply and speculative labor demand.

Understanding software and software production has always been a moving mark, difficult to pinpoint, and thus difficult to prescribe relevant and contemporary economic
development policy. One contribution of this dissertation is to track the changing nature of software as it relates to the form of the software itself, the form of software labor, and the form of software as a commodity. These aspects of software are molded by new developments in hardware, new advances in software languages and libraries, new forms of software development, changing firm strategies, and the changing spatial organization of production.

Early software workers flipped binary switches on machines to enter instructions and now software workers type code and have software libraries or machine learning develop code based on constraints. During this transformation, software has gone from being tightly coupled with the computing machines in the form of switches, to almost universal applications that share platforms. Software’s form as a commodity evolved alongside these changes from custom one-off construction like developments, to continuous development models based on beta code or proof-of-concept developments. Physically, the places software operates has evolved from enormous mechanical binary switches, to punch cards, to tape, to optical disks, to minuscule flash memory – from binary switches, to 3D NAND switches. Software workers have had to stay abreast of the changes, reacting to new developments to retain their skilled status. In doing so, workers themselves seek to improve their own efficiency, competing with other workers, and even producing software whichdeskills or automates the work of other workers.

Perhaps one of the most confusing and difficult parts of researching and understanding software labor comes methodologically. Is software production an industry? Certainly. But software production is also more than an industry – as I show in the dissertation, there are zero industries (using either NAICS of SIC methods of industry classification) that do not have software workers. That software workers have permeated into all areas of production, unlike, as an example, logging or restaurant server which are found almost exclusively in natural resource industries and eating and drinking places respectively.
Thus, software production is more akin to management and administration than industry-specific occupations, forcing scholars to examine software production as a cross-industry laboring activity. The free access to open-source software also confuses traditional examinations of software production because the product, or output, of large portions of software labor is given away freely, only then to sell access to complementary software services, supplementary software products, or licensed access to free software running on advanced computers. Cloud computing and advanced internet infrastructure make this possible in ways previously thought unimaginable.

Regional variation and divergence only exacerbate our ability to understand software production. “Modes” of software production, characterized by the configuration of a region’s software industrial district, undergo periods of divergence and normalization. Silicon Valley’s concentration of software production continues to be an industry- and production-setting region, setting standards and practices that other firms and workers mimic. Silicon Valley is unique and other regions have differing industrial district formations that result in different modes of software production. Software production in second-tier regions – regions competing with Silicon Valley and contributing to standards and practices in their own right – diverge from dominant forms of software production in each software period.

Second-tier regions like Seattle, Portland, Austin, Boston, and Research Triangle all have large agglomerations of software production activity, and produce software under varying social relations, varying institutional relations, and varying forces of production. Regional divergence results from regional firms competing and experimenting with new commodities, technologies, organization, and labor practices. The needs of global capital, technological infrastructure, and chance determine which software technologies become dominant in the next software period, producing regional winners and losers. The resulting variegation – the variation of the state of regional production in both
outward appearance and internal structure – provide opportunity between each software commodity period for a new region to emerge as dominant trend-setter. The high-tech production and specialization literature classically describes Boston as the almost Silicon Valley; highlighting the importance of the evolutionary nature of regions and their industrial districts, Boston is by far a loser. Yet, to date, the evolutionary nature and industrial district formations of software production remain understudied.

Software production is widely perceived to be a source of good quality jobs and a pathway of opportunity underscored by President Obama’s media stint. Yet, the industrial district formations of regionally specific modes of production yield significant variation in wage distributions. These variations are particularly evident along lines of gender and race/ethnicity, calling into question the relationship between regionally-specific modes of software production and the ability to produce wealth for traditionally marginalized groups like women and persons of color. Further, these variations suggest that the changing nature of software production must tracked to understand the structural pressures on firms and software workers which hinder the ability of regions to produce wealth and for traditionally marginalized workers to find jobs that lead to successful careers. As I show, software workers, perceived to have highly sought after and safe jobs, are hyper sensitive to deskilling. Further, the occupational bifurcation of software jobs leads to dualized, segmented labor markets of high-skilled software developers and low-skilled software producers-users characterized by repetitive tasks.

The structural forces of deskilling and segmentation take on local variations because of the regional modes of production. As part of the regional modes of production, local institutions and labor groups seek mediate and mitigate the structural forces so that software production can both produce wealth and increase equity. The power and success of local institutions connect to the changing nature of software and the industrial district formation. That is, the priorities of local institutions are evident by
the structure of software in the region. In Portland and Austin, as I show, workers are hypersensitive to automation and deskilling and thus workers seek to organize to combat the inequitable impacts of these forces. In Portland, workers leverage their inter-firm, inter-industry connections while in Austin, workers seek to embolden a startup community by encouraging diversity in programming education and access. These examples stress that labor organizing, whether formally as a union or informally in events or initiatives, is vital for the generation wealth for workers.

The purpose of this mixed-methods dissertation is to understand the dynamics of software production in regional economies. The first set of research questions, addressed in Chapter 4, focuses on understanding the extent of software production and cataloging the changing nature of software since 1970. Combining historical research, a large dataset of software occupations across the US, and interview data, Chapter 4 uses four periods of software commodities to look at the changing nature of software labor while software technologies are actively developed.

Chapter 5 then builds on the findings from the Chapter 4. Chapter 4 argues, among other things, that software must be measured by looking at software occupations. This ambitious chapter catalogs regional changes in software production since 1970 and examines software production in three case study cities of Austin, Texas, Portland, Oregon, and Seattle, Washington. Lastly, Chapter 5 connects the local form of software production to the variations in income distributions and occupational structure within software work itself.

Finally, Chapter 6 examines the structural pressures on software labor markets produces inequity along lines of gender and race/ethnicity. This chapter groups occupations into labor market tiers, then looks at incomes and inclusion rates for women and minorities. Lastly, Chapter 6 uses interviews to examine ways the adverse pressures on software labor markets are mediated, mitigated, and challenged by local communities and
institutions.

One core argument of this research is that software fundamentally is misunderstood as a laboring activity. More precisely, software is unique in that the output of laboring software workers is software – which may potentially automate the software worker that created it. I argue, throughout the three substantive chapters, that this necessitates greater expediency in prioritizing collective action and addressing equity issues in software production.

In Chapter 2, I review the necessary literature, documenting examples of high-tech regional production, labor market dynamics, and a conceptual framework for understanding software production. Chapter 3 then documents my research design and research method. Chapters 4 through 6 I outlined above. In Chapter 7, I review my findings and provide some concluding thoughts. Appendix A documents the method used to measure software labor at the metropolitan level, a contribution to the literature by itself. Appendix B contains supplementary data which are too numerous to include in the main part of the dissertation.

Theoretically, this research offers insight into how this current moment of software production differs from other moments of production (Fordist, post-Fordist/flexible accumulation) and the importance of certain kinds of regionally specific modes of production that might yield more desirable industry and regional economic development outcomes. To capture these desirable economic development outcomes, this research provides relevant economic development policy directions. These understandings engage in providing new empirical case studies and expand existing theoretical understandings regarding capitalist production under the new relationships of cognitive-cultural capitalism.

Let’s begin.
2 Literature Review

2.1 Framework for Regional Analysis of Software Production

Understanding the contemporary landscape of software production requires us to grapple with both the enduring and changing character of technology under capitalism itself. Thus, this research speaks to a cross-disciplinary interest between economic development, industrial geography, and labor market scholars. This fruitful line of inquiry continues to generate theories of regional evolution and high-tech production. Addressing the specifics of software production and the changing nature of software production, I incorporate a theoretical approach with four main dimensions: profit cycles, capitalist variants, industrial districts typologies, and software periods.

Fortunately, Campbell-Kelly (2004) periodizes software production between 1950 and 1995, introducing three software periods that span about 15 years each. Contract software starting in the mid 1950s, corporate software starting in the mid 1960s, and mass-market software starting in 1980. Campbell-Kelly and Garcia-Swartz (2015) later adds a fourth period starting in 1995 called the internet software period. This useful periodization characterizes dominant technologies and forms of software since computing and software’s development. A problem with this particular form of periodization lies in the choice of transition years, suggesting two changes. First, the US military was leveraging software contracting as computing services prior to 1956, as documented in the book. Second, the transition years after the initial period mark the first time that
technologies were developed – technologies that then went on to characterize the period.

History is read by the winners, and if we applied this method, we would lose sight of the fact that there were many competing technologies fighting to characterize the next period. For example, the mass-market period could have been the artificial intelligence period had developments in fifth generation languages taken off. This is also true of the fourth period added later, called the internet, which starts in 1995. The internet still exhibited evidence of mass-market software, as the internet was still a platform for purchasing to run locally. Instead, considering transition periods, or declines in technologies, scholars might look at when different forms of software became popular and acknowledge the decline of other forms of software. For example, how do we think about the fact that Apple’s App Store allows Mac users to purchase software? In the mass-market software period (third), software did connect over the internet, but in the cloud and mobile period (a revised fourth), software runs over the internet and requires connection. An example of this is Google Docs or Gmail. What we’ve seen as the internet since roughly 2004 when Google released GMail: a web 2.0 period where websites are interactive.

A form of periodization that may prove useful is that of the five profit cycles. Markusen contributes a theory of how industry restructures in unison with changes in the profit cycle starting first from zero profit, to monopoly profit, normal profit, declining profit, and finally negative profit. In each of the product stages, competition changes the nature of the industry. At first, historical accident best describes the spatial configuration, but upon innovation, these firms tend to co-locate. Upon standardization, industry undergoes cost-cutting firm consolidation, avoids unionized regions, and seek other features to cut costs in production (Markusen 1986). Markusen (1986) explained the complexity of locational choices based on profit cycle, noting that the importance of research and development, suppliers, standardization, and labor costs changed as the
product matured between five different profit cycles.

Regions dominated by high technology, innovative firms, such as California’s Silicon Valley, will have relatively bifurcated labor force with large proportions of professional and technical workers and relatively unskilled assembly workers. Regions dominated by mature oligopolistic industrial sectors will have large proportions of unionized, skilled blue-collar workers. Profit-squeezed sectors, oligopolized or not, will migrate to areas with low-cost, unorganized labor. (Markusen 1986, 10)

The pursuit of low-cost unorganized labor is in part what drives industries to seek “locational windows” termed by (Storper and Walker 1989b) to describe the move to a new region that may provide the reconfiguration necessary for the industry to begin a new profit cycle. Markusen, DiGiovanna, and Y. S. Lee (1999) demonstrates that different regional modes of production yield lasting differential and broad-based regional economic development outcomes and that regional modes of production can analyzed by uncovering each regions software industrial district. A succinct typology of the industrial districts – spatial configurations of these sectors – are outlined in later work: Marshallian districts consist of large numbers of locally networked smaller firms; hub and spoke districts revolve around a single large firm and many smaller firms; and satellite platform districts are unconnected satellite offices of large firms connected to their respective headquarters and other satellite offices (Markusen 1996).

The relocation of semiconductor and computer manufacturing to Silicon Valley is an example of a capitalized locational window. The relocation and success of Silicon Valley’s semiconductor and computer manufacturing depended on the cost cutting measures afforded by the region’s strong relationship between university and industry. Silicon Valley was a Marshallian industrial district of semiconductor and computer manufacturing
that benefited from the knowledge spillover of the university, the flexible and specialized labor of the region, industrial capacity, and informal networking (Saxenian 1996). Later work also advocates for examining “second-tier regions” which challenge the primacy of places like Silicon Valley. Here the notion of locational window proves useful as second-tier regions may provide locational windows during shifts in the profit cycle. I expand on the districts later, but compile characteristics on each industrial district in Table 2.1, Table 2.2, and Table 2.3, and include reproductions of the visuals in Figure 2.1, Figure 2.2, Figure 2.3, and Figure 2.4.

A different periodization seeks to understand broad shifts in capitalist industrial logics, declaring the oncoming of global capitalism and the decline of monopoly capitalism (J. Graham et al. 1988). A summary table is provided later in Figure 4.2. The framework is Marxian in nature and focuses on inter- and intra-class relationships. Each period – or capitalist variant – describes relationships thus describes sets of factors that describe each set of relationships. For capital-labor (inter-class), these metrics are the use of technology, the logic of the labor process, and the form of the reserve army. For capital-capital (intra-class), these metrics are the form of centralization, the degree of concentration, and the form of international capital circuits. Global capital is marked by shifts toward increased computerization, small-batch specialty production, non-unionized labor, capital diversification, increased capital concentration, and international finance capital. This describes an economy-wide crisis of production rather than an industry-wide logics, contrasting with Markusen (1986). Notably, J. Graham et al. (1988) gives us a method for analyzing regions – by examining new forms of technology, existing labor practices, and firm organization.

During each software period, software takes on a new commodity form. Thus, the production of software undergoes each of the five profit cycles associated with a product cycle. As a new commodity form gains prominence, the share of the market
share of the commodity form decreases. In software, this is typically the case because new product form of software also grows the entire size of the market. During each of software's historical commodity periods, software production has taken on characteristics of the larger economy-wide changes in the economy. For example, the contracting software (from software periods) product cycle (from profit cycles) had capital-capital and capital-labor relationships mimicking the economy-wide class relationships of competitive capitalism (capitalist variants). Each of these sets of periods has particular sets urban spatial arrangements (from industrial districts) which maximizes software period, product cycle, and capitalist variant relationships.

2.2 Labor Markets and the Labor Process

2.2.1 Structure and wage distributions

Labor market theorists attempt to understand how skills, job matching, and career development impact regional economies. Further, they seek to understand the points for market intervention, acknowledging the problems with describing labor as a market. This section builds from the work of Braverman (1998), Chris Tilly and Charles Tilly (1997), and Benner, Leete, and Pastor (2007).

Braverman (1998) claims that the general law of capitalist division of labor is to reduce the necessary specialized knowledge and training of laborers so that the majority of laborers perform simple, often repetitive laboring tasks. The result is an ever decreasing number of specialized occupations and large number of occupations which require minimal training. The labor process is then structured around this occupational bifurcation, as dictated by the needs of capitalist production.

This labor process structure produces significant disparity in wage distributions. The work of Mincer (1992) characterizes general themes of labor economics as the study of
wage profiles and distributions, labor mobility, and the relationship between technology and labor. The human capital model serves as a more expansive lens for viewing labor economics compared to traditional views, which ignore the skills and training investments that return back into the system.

If the current growth of demand for human capital is based on skill-biased technology, and skills acquired at school and on the job are a function of the quality of learning and not merely of the time spent in it, a bottleneck in the expansion of human capital supplies may lie in the inadequate quality of learning absorbed by the work force, especially at the elementary and secondary levels of schooling. (Mincer 1992, 135)

Mincer (1992) finds, first, that greater amounts of education obtained by a worker correlates to a greater wage, and that education reduces risk of unemployment. Second, younger workers tend to be more geographically mobile since the payoff becomes higher over time. Third, while some human capital is transferable between firms, some human capital is firm-specific. This can be both a deterrent and an impetus for inter-firm labor mobility. Fourth, technology tends to steepen wage profiles of labor. These broad themes are impacted by the supply and demand curves of labor and human capital. Shifts in the demand curve can be caused by changes in public policies, public levels of education, and overall family income levels.

Chris Tilly and Charles Tilly (1997) expand and formalize the ideas of Mincer in their articulation of institutionalism, a framework of thought that directs labor markets to look at social context. Social norms and transactions between actors form the base of this framework, creating a network and hierarchy of labor market actors. Labor markets exhibit combinations of incentive, contract, and embeddedness forms of labor mechanisms. These labor mechanisms exert, and control, power. Workers, employers,
and institutions navigate these labor mechanisms to pursue their own objectives, often navigating the social, technological, and government constraints placed on labor markets. These objectives are multi-fold, but the contract between these actors can include various aspects of the incentive labor mechanism. These contracts between actors can be brokered through social interactions in labor hierarchies, coalitions, and markets. The state plays a role in labor markets, but the involvement of the state can be different since government objectives are often changing with social norms. Chris Tilly and Charles Tilly (1997) describe the commodification of everyday life (proletarianization) in US history, noting the change in labor from work in households to work that adds exchange value.

Social relations, as well as economic conditions, channel history’s impact on technological choices. This is visible in the innovation process itself. ... More broadly, social relations, power, and culture in the workforce ... set the frame for efficiency. (Chris Tilly and Charles Tilly 1997, 107)

Chris Tilly and Charles Tilly (1997) assert that this can be observed by examining cultural history. Labor markets, and the relationship between actors in the market, have changed as a result of the interests of capital, which are often but not always, expressed through employers and firms.

2.2.2 Employer and firm dynamics

This change is cataloged by Osterman (1999), asserting that the institutions of the labor market in the US post-war period have been undermined to shift and manage risk.

The wage structure has been shattered, and market forces have much greater impact on compensation than they have had in the past. (Osterman 1999, 68)
Existing views of the labor market share an underlying premise of a long-term contract between worker and employer. This model does not allow for the natural ebbs and flows of the economy, limiting the contract to adapt to changing economic conditions. Corporate governance, the government’s role as an intermediary, and relationships between industrial labor actors helped shape these constraints. Firms’ ever present profit motive combined with increased global competition, new technological advancements, and firm reorganization the underlying long-term contract commitment that undergird secure employment structures are dissipating. Osterman provides several longitudinal studies to document this dissipation, measured in layoffs, quits, and tenure, among other criteria. As Kalleberg (2009) shows, temporary work is one way in which both employers and workers have shifted risk to deal with economic tides. Chris Tilly and Charles Tilly also introduce bounded rationality into the labor market and job market. Chris Tilly and Charles Tilly suggest that the classical markets are less important than economists may suggest, pointing to the fact that most work is found through networks. This further muddles the supply and demand curves in labor markets and continues to add to their divergences in distributions. History, race, class, and gender are all part of networks that circumvent traditional methods of employment.

Benner (2003) asserts three factors – heightened global competition, firm reorganization, and technological advancement – have changed the relationship between worker and employer. Labor markets have been adapting to these factors in part with institutions that act as labor-market intermediaries between both ends of the labor contract. This ultimately reduces transactions costs, building networks and managing risk. These private, public, and membership-based intermediaries increase the reflexivity of labor markets to match the reflexivity of firms, both spatially located, which contributes to regional economic development Benner (2003) and Benner, Leete, and Pastor (2007).
During the 1990s it was common for observers to see in the rapidly growing information technology sector a paradigm for the future economy. The eventual (and in fact entirely predictable) failure of the sector to sustain its growth path, and indeed its partial collapse in the early twenty-first century, makes it possible to view the changes it introduced in a more balanced perspective. (Crouch 2005, 111)

Crouch (2005) asserts that skills are a collective good, not just an individual characteristic, since it is the employers that use the skills. Crouch also surveys the techniques from which skills can be built: the government, the labor market (experience), corporate hierarchies, associations, communities, and networks. Each of these techniques is not a system for building skills, instead they are individual sources that make up part of a larger system of skill formation. The institutions and systems have encountered change from concentrations of skilled labor and the globalizing labor market.

2.2.3 Skill-building

Technology improvements have required firms to adapt to both specialization and globalization, which may be the source of the changes in skill-formation institutions. Job training is an important part of the labor market since job training is a form of a career ladder or pathway and an investment in human capital.

Operationally, the trend over the past three decades has been to devolve authority for [workforce development] program design and service delivery from the federal government to the states or localities. ... policymakers generally agree that workforce development programs and policies should allow flexibility to respond to local economic needs and business cycles. (Barnow and Nightingale 2007, 36)
The sources of job training and the responsibility of job training are in flux. Scholars cite nuanced reasons for the ineffectiveness of job training, acknowledging the hardships of disadvantaged populations and the general lack of effective policy. Each argument is framed in the larger context of a changing labor market – in which politicians are cutting funding for workforce development programs and labor institutions are emerging to fill the vacuum left behind by retreating corporations and programs (Barnow and Nightingale 2007).

Labor markets exhibit queuing behaviors. Thurow (1975) asserts that there is a queue of jobs within the labor market, both from within and between primary and secondary markets. Thurow further asserts that skills are acquired on the job, which means the labor market is not a market for trained labor; rather, it is a labor market for trainable labor. Trained labor supply depends on the trained labor demand, thus the supply and demand are not independent of one another. Education acquisition is a leading method for workers to demonstrate their trainability – or background characteristics that show ability to observe professional group norms. The labor market can be viewed as both a job queue and a distribution of job opportunities to grab a holistic view of the market. Reskin and Roos (1990) expand on Thurow’s labor queue model, providing clarity in the distribution of job opportunities and incomes by describing a job queue. Where the labor queue models workers in rank of attractiveness, the job queue models jobs in rank of attractiveness.

Often technological changes that elaborated the division of labor, deskill work, or altered working conditions set the stage for occupational decline. Normally, jobs change because employers transform the technology of production or reorganize the work. ... as the computer workforce grew, managers sought to contain labor costs by separating from programming two new oc-
ocupations: systems analysts, who designed information systems; and coders, who translated programs into computer codes and entered data. (Reskin and Roos 1990, 43)

Both queues generate the uneven distribution of jobs, resulting in the best jobs being taken by best workers and the worst jobs being taken by bottom-ranked workers. This also means some bottom-ranked workers are unemployed.

2.2.4 Gender and workforce development

Historically, positions of men and women with the same background characteristics have not been even in the labor queue due to the customs of sex-based hiring. Ignoring sex in the labor queue undermines the high rankings of male employers and managers.

... occupational sex segregation has been more resistant to change than race segregation. (Reskin and Roos 1990, 6)

There are many conditions that allow in-roads for women into male-dominated positions, generally involving reranking positions by males and females alike or a reranking of the position by the employer around gender-based stereotypes. Most feminizing positions have involved emotional labor, requiring the generation or suppression of emotions. However, all in-roads adhere to both a labor queue and a job queue – ranking of both workers and jobs. Reskin and Roos also importantly document various industries that undergo feminization and the different nature and structure of feminization. The feminization of a labor market, characterized by the growth of women’s share of production, may produce meaningful careers for women. Some forms of feminization reinforce broader gender and racial dynamics by allowing women and minorities at only the bottom of the labor market.
Firms are not necessarily vested in workforce development when there are other ways of finding or maintaining labor. But the effects of a globalizing workforce (labor market) are scarcely touched upon. For example, Barnow and Nightingale (2007) claim that firms cannot find enough skilled labor. The crux of their argument relies on this assumption. They assert that maintaining a skilled workforce involves educating, training, and adapting existing workforces. Due to severe public-funding cuts, most of these training efforts are privately funded. Thus, in order to meet the skill-requirement demands of firms, policy must seek to maintain or improve worker productivity through increasing skills, improving worker security and safety, improving employment opportunities, and rewarding strong work ethic. Barnow and Nightingale survey six policy categories, each of which sees the worker as a customer which must be served. Barnow and Nightingale (2007) close with the thought that the Bush administration is shifting away from programs and institutions to individuals, and instead focusing on incentivizing businesses to train workers. Even this still sees the worker as the customer, but simply transfers the responsibility to the firm. Barnow and Nightingale (2007) additionally acknowledge the importance of local policy to supplement federal- or state-level policy gaps.

Promoting advancement in the labor market for low-wage workers requires changes in firms and in workforce delivery systems. (Giloth 2000, 346)

Giloth, however, notes that the failures of some workforce development training programs are due to disengaged employers. Giloth insists on substantial employer engagement, deep community connections, career advancement, integrative human service supports, contextual and industry-driven education and training, reformed community colleges, and the connective tissue of networks. No matter the labor market, regions (not arbitrary municipalities) and people (e.g., race matters) are the right levels at which to affect change. Giloth finally notes the important role of the labor-market intermediary.
Holzer and Poverty (2008) attempts to understand the failures of workforce development policy. Holzer and Poverty note that skills are determinants of labor market earnings, the US continues to cut workforce development programs. Workforce development programs, when done right, are cost-effective. Additionally, workforce development programs for disadvantaged populations have decreased due to a lack of voice for the rights of the poor, which means these cuts are largely political. Workforce development practitioners and scholars see career pathways and ladders as important forms of workforce development.

2.2.5 Labor market intermediaries

There are two other aspects that must be considered, namely the context in which the intermediary emerges. Career pathways are well established in Boston hospitals, for example, which explains their high employee-retention rates in comparison to those that don’t participate in career pathway programs (Fitzgerald 2006). Benner (2003) documents types of service roles that intermediaries can take to shape labor markets. Using three different broad categories of roles – meeting, molding, and making – Benner (2003) describes how a labor market intermediary (or specifically a workforce development intermediary) might use these roles to impact the labor market.

Workforce development, career ladders, working with LMIs, and other forms of labor involvement are all high-road relationships between industries, firms, and labor. Yet, industry pressures can result in disinvestments in labor. Here, Bernhardt, Dresser, and Rogers (2001) describes the cycles and incentives of firms and industries to pursue labor relationships outside the high-road relationships examined earlier. In the absence of a partnership between labor and capital, low-road relationships may more broadly occur. The success of instituting government-sponsored partnerships to take a “German” approach to promoting high-road paths for corporations. Bernhardt, Dresser, and Rogers also clearly described the risks that capital incurs when choosing the high-road in the
absence of an agreement or partnership. Bernhardt, Dresser, and Rogers assert that the successful implementation of partnerships in Milwaukie’s manufacturing sector could be taken to any region seeking economic development through workforce development. In fact, three items are suggested and outlined for practitioners.

Changing technology, global competition, and a host of other factors have led to stagnant wages on the lower rungs of the economic ladder and limited opportunities for skill building and advancement. The sectoral approach to workforce development offers strategies to improve outcomes for low-income workers that are based on understanding the labor market system within a particular industry and region and developing tactics that both benefit workers and make economic sense in today’s business environment. (Conway, Dworak-Muñoz, and Blair 2004, 2)

This fundamental understanding helped to frame the descriptions and plans set forth by other scholars. Conway, Dworak-Muñoz, and Blair (2004), Conway (2007), and Schrock (2011) discuss labor relationships made through these kinds of partnerships, both formal and informal. This work supports high-road firm behavior while providing incentives external outside of the aims of economic development. This work promotes firm vitality by creating firms networks and forging collaborative relationships between capital and labor. Comparing Schrock’s analysis from Chicago to Bernhardt, Dresser, and Rogers’s analysis in Milwaukie reveals striking similarities. “Demand-driven” and “identifying a sector with the most immediate payoff” are two sides of one coin – they aim to find the sector that demands aid through lackluster performance. This leads to “sector-focused” interventions, which focus efforts on one industry. System-integration is the core of Bernhardt, Dresser, and Rogers’s industrious partnerships that bring capital and labor together.
Previous empirical research on the impact of business assistance living wage laws has detected significant decreases in employment. Yet experts have questioned this past research on the grounds that the data sources could not detect urban level impacts and that they did not adequately control for whether cities actually enforce their business assistance provisions. (Lester and Jacobs 2010, 28)

Outside of knowledge workers, the demand for the sustainability of wages is expanded by Bartik (2004)’s description of local living wages. This framing is useful to shine a light on the both the differences between local and federal living-wage laws and the components necessary to create successful local living-wage requirements. Lester and Jacobs (2010) asserts that wage minimums have no negative effect on city-wide employment levels and do not harm low-wage workers. Lester and Jacobs (2010) claim to provide one of the most methodologically sound studies to date for quantifying the impacts of minimum- and living-wage laws. While it is unclear how different cities might change based on the outcomes of Lester and Jacobs, this is strong evidence that counters observations of other scholars, notably Adams and Neumark (2004). Lastly, Bartik laid out the argument for federal minimum-wage standards, even if it wasn’t intended.

2.3 Regional Development and Industrial Restructuring

2.3.1 Restructuring spurs research

Understanding regional economies and regional development outcomes has always been closely linked to firms’ organizational and technological innovations. After the post-war growth period, the increased fungibility of both capital and labor as part of broad political economy transformations became tightly linked to a decline in US manufacturing jobs. Under new international pressures, firms sought organizational efficiency through out-
sourcing and offshoring and sought production efficiency through technological advances (Bluestone and Harrison 1982). While manufacturing industries underwent a restructuring that unevenly impacted regional economies, new high-tech industries brought promise for regional development (Piore and Sabel 1986). The resurgent importance of understanding regional economies required an understanding of the uneven decline of American manufacturing industries, the growth of new industrial production in Asia, the regional topology of new high-tech industries, and the tendency for some regions to incorporate high-tech industries into their economic base (Storper 1997).

Paradigms from traditional economics, evolutionary economics, and geography attempted to address the changing spatial nature of capital, labor, and innovation. Traditional economists, like Losch (1954), Isard (1960), and Christaller (1966), incorporated space into equilibrium models to develop input-output models based on location theory. Evolutionary economists, like Nelson and Winter (1985) and Antonelli (2003), incorporated technological path-dependence into regional models. In doing so, they considered technological change as endogenous to economic systems, showing empirically that technological spillover takes a spatial pattern and is directly linked to regional innovations by firms. Heterodox scholars, like Massey and Meegan (1978) and Bluestone and Harrison (1982), examined regions as “relatively passive subjects of translocal industries” (Markusen, DiGiovanna, and Y. S. Lee 1999, 43). Insights from research on Third Italy led yet another wave of research on business milieu and cooperative production networks by scholars, like Piore and Sabel (1986) and later Saxenian (1996).

2.3.2 Heterodox paradigms: regional forms of accumulation and production

Amid economic geographers, a heterodox paradigm emerged from numerous works (Hall, Markusen, et al. 1983; Hall and Markusen 1985; Markusen 1986; Storper and Walker 1989b; Markusen, Hall, Campbell, et al. 1991; Sayer and Walker 1992; Scott and Storper
1992; Amin 1994; Saxenian 1996). The heterodox paradigm rests on two foundational components. The first component is what Storper calls the “holy trinity” of organizations, territories, and technology, and the relationships between them. In this trinity, organizations primarily include firms but are also defined by the institutions and groups that link firms together. Technologies, or more specifically new technologies, involve production innovations. Territories are the places in which local interactions between actors happen and the spillovers of technologies and organizational patterns are shared. And yet changes in the “overall nature of ... capitalism” requires that understandings about regions and their economies consider the economy as a set of relations, the economic processes as conversation and coordination, and the subjects as reflexive. Accumulation then can be understood not as material accumulation but instead as relational accumulation. Regions house accumulations of relationships in which firms compete and cooperate (Storper 1997).

This notion of relational accumulation, competition, and cooperation lends itself to the second foundational component: that of disequilibrium, rather than equilibrium, growth. The spatial heterogeneity of labor markets and the increased mobility of capital and capital stock exacerbate regional and labor competition, producing uneven regional and industrial development (Walker 1978). This uneven regional and industrial development undergirds a tendency toward capital accumulation through disequilibrium: “thus, territorial extension, differentiation, and instability are not afflictions visited on industrialization but conditions upon which capitalist development operates.” New industrial territories and new industrial complexes, based on organizational and technological innovations in production, generate higher rates of profit. These new territories and complexes attract labor and capital that generate regional growth – reifying the uneven regional and industrial development from which they arose. This model relies on the need to realize profits quickly (cyclically: investment, production, realization, new investment)
because of hyper competition and shrinking profits from organizational and technological innovations. Temporary geographic breaks in industrial barriers to entry present opportunities for innovative firms and industries to territorially expand in pursuit of higher profit returns. In summary, geographical industrialization, alongside organizational and technological innovations, can be a form of territorial capitalism resting on place-specific regimes of accumulation and production, or industrial districts. The firms in these industrial districts are protective of their territory, technological innovations, and labor markets, relying on cultural, social, and political institutions to create boundaries (Storper and Walker 1989a).

2.3.3 Studying regions: Firms, labor markets, place and space

As the crisis of fordism deepened over the 1970s, geographers and regional scientists began to produce deeply pessimistic accounts of the corresponding quandaries that were then proliferating in the principal cities and regions of that period. (Scott 2011a)

The heterodox paradigm of regional economic development was employed to study both declining manufacturing regions and rising high-tech silicon manufacturing regions by examining firms and the labor markets that workers constituted.

Capitalist firms constitute the major actors and decision makers in industrial districts. Non-firm actors, government institutions, social organizations, and place-specific histories and cultures condition the terms of the firm and labor behaviors. And while “regional performance is often inferred as a function of the behavior and dynamics of certain key firms or industries ... [however] Studying industrial structure with secondary data can guard against concentrating overly much on faddish industries” (Markusen 1986, 46). Industries can be selected by observing the primary contributors to regional economies.
As we know from Saxenian, the success of Silicon Valley in the 1980s can be attributed to the electronics industry’s production flexibility, overseas relationships with manufacturers, and linkages to university R&D. However, other key regional industries were connected to the electronics industry, including the aerospace industry (Saxenian 1983; Saxenian Fall 1990; Saxenian 1996). Organizational and technological innovations came not just from firm flexibility, firm networks, and university R&D, but also came from the human capital cross-over from other industries and the capacity for these innovations to spread from one firm to another.

A major contribution of this research is that while the geographic strategies of firms shape the character and vigor of regional economies (Markusen, DiGiovanna, and Y. S. Lee 1999), so too does human capital, the place-specific history (from uneven development), and the space-specific urban form and built environment. I provide a more explicit framework in the next section. While firms are protective of the labor and technological boundaries, firms also take advantage of these boundaries and their heterogeneity to exert control. Firms can use their power within regions through threats of departure or replacement (Storper and Harrison 1991). In the case of labor boundaries, firms view labor as “purchased” on future performance and its reproducibility rather than on current output. Within the corporation, firms can use the threat of switching plants to minimize the chance of labor organizing. High-skilled workers may also take advantage of this spatial heterogeneity by placing demands for their spatially rare or in-demand skill-set. These insights only further stress the importance of understanding regional labor markets and the spaces and places where labor and firms interact as shaping regional economies.

2.3.4 From post-Fordism to cognitive-cultural capitalism

The industrial district literature is most well known for studying the rapid rise of darling Silicon Valley exemplified by Saxenian. The basis of this work, and the immense literature
it spawned, is manufacturing-based production. The work of (Saxenian 1983) focuses explicitly on silicon-chip production. This work spurred scholars’ interest in the emergence of “new successful forms of production” that were regionally specific (Storper 1997, 3), and yet, even this new literature focused on manufacturing-based industries, like biotech (Cortright and Mayer 2002; Feser 1998; Malizia and Feser 1999), the military-industrial complex (Markusen, Hall, Campbell, et al. 1991), or other high-tech hardware production (Chapple et al. 2004; Cortright and Mayer 2001; Mayer 2012). A key finding common to all this research was that technological innovation was not necessarily physically and geographically located in the same places as production and that vertical and horizontal disintegration had a spatial component. This contributed to the capacity for firms and industries to have greater organizational flexibility, leaving room for new organizational new innovation in the production process. These findings were emblematic of broader changes in the nature of capitalism, which scholars attributed to a renegotiation of capitalist production relationships. This broad shift was sometimes called post-Fordism (MacDonald 1991), flexible specialization (Piore and Sabel 1986), and flexible accumulation (Harvey 1987). At risk of oversimplifying the nuances between these authors’ frameworks, at their core they all a technologically-based economic paradigm supported by social and institutional frameworks for spatial ordering (Amin 1994). For simplicity, I will refer to these overarching political economy changes in capitalism as post-Fordism and in firm-level strategies as flexible specialization.

With the exception of Hall, Markusen, et al. (1983) and Colclough and Tolbert (1992), software production remains largely under-researched. Storper and Walker (1989b) refer to a burgeoning “information industry” yet their imaginations are limited to the manufacturing-based production of information-processing equipment that may replace silicon – similar, perhaps, to the now-defunct textile industry. Contrary to this gap, software’s importance has only increased.
In very schematic terms, we might recognize three broad historical phases of capitalist development, based on their characteristic technologies, leading sectors, labor market relations and competitive dynamics. Equally schematically, we can also distinguish three corresponding waves of urbanization. (Scott 2011b)

This quote helps illustrate the importance of studying software and regions together; software is a leading production activity and regions are critical parts of third-wave urbanization. And yet, as employment in software production grows alongside software employment in non-software firms, we still do not know the specificities of the various regionally specific modes of software production. In fact, just information-technology-related industries comprise 10% or more of 15 metropolitan areas with total populations over 1 million people. More importantly, other industries and other types of production are beginning to look a lot more like software. Recent research shows the number of non-software industries with software workers is rapidly growing including finance, oil & gas, and manufacturing (Mahmoudi and Schrock, in progress).

Another major contribution of this work is to assert that while existing forms of industrial production exhibit characteristics of post-Fordism, software and software-based production exhibit characteristics of cognitive-cultural capitalism. The fact that other industries continue to look more like software production only stresses the importance. The framework of cognitive-cultural capitalism (CCC) builds off of the academic heritage of post-Fordism but employs updated observations of the nature of capital. CCC can be defined as having three primary relationships:

(1) the new forces of production that reside in digital technologies of computing and communication; (2) the new divisions of labor that are appearing in the detailed organization of production and in related processes of social
re-stratification, and (3) the intensifying role of mental and affective human assets (alternatively, cognition and culture) in the commodity production system at large. (Scott 2011a)

Software fits closely into this paradigm as an inherently digital form of production and enmeshed politics of class and education, almost exclusively resting on cognition as its form of production.

Of particular importance in software is, of course, the mode and process of production. Software requires high skill-levels and its reliance on cognition as labor software programming means that software process production are nearly entirely changes in spatial orientation or changes in the labor process. As is well established in manufacturing, additional workers generally translate into additional productivity. In software, because the production process is cognitive, the way in which additional workers interact – or collaborate – dictates the level of productivity rather than the presence of capital machinery. Thus, many innovations happen during the process of production and the type of network, whether informal or formal, becomes objects of interest for firms (Camagni 1991; Chesbrough and Appleyard 2007; Mayer 2012; Bessant et al. 2012). The role of the city or region as a paradigm for network production processes is also of interest, particularly in the configuration of space to maximize interaction, innovation, and productivity (Camagni 1991; Camagni 1993; Howells 2002; Simmie 2003) and their impacts on the region as a whole in incomes and the capacity to recover from economic shock (Simmie 2003; Lester and Jacobs 2010). The production and innovation overlap suggests that the regional innovation systems are intertwined with regional modes of production as industrial districts (B. T. Asheim 2000; B. Asheim 2007). With these changes in production and innovation, the idea that industries have spatial and social characteristics that affect regional development outcomes seems to gain new importance.
After all, Storper and Walker claimed that industries produce regions (Storper and Walker 1989a) and Harvey noted that “the history of urbanization of capital is at least in part a history of its evolving labor market geography” (Harvey 1987, 19).

2.4 Existing Research on Regional High-Tech Production

Several important cases examining the relationship between high-tech industries and regions are of key importance. Of most importance, partly due to its popularity and the archetypal status of Silicon Valley, is the work of Saxenian.

Route 128’s independent-firm-based system had provided economic scale and organizational stability that were valuable in an earlier era, by the 1980s they served primarily to discourage adaptation. The commitment of local companies to vertical integration meant that technical capabilities and know-how . . . remained locked up within large firms. The paucity of horizontal communications stifled opportunities for experimentation and learning while traditional corporate structures limited the development of managerial initiative and skill . . . This may have imposed a minor inconvenience to large firms, it became a significant disadvantage for start-ups and small firms that were unable to learn about or acquire state-of-the art components or services as rapidly as their West Coast counterparts. (Saxenian 1996, 115)

Saxenian (1996) compared the explosive growth of the electronics and semiconductor industries in California’s Silicon Valley with the decline of similar industries along Route 128 in Massachusetts. Saxenian’s in-depth analysis, put simply, found that the rate of innovation in Silicon Valley was higher than that of Route 128. This competitive advantage came from the dense communications networks and the transfer of ideas across firm boundaries.
The flexible milieu of Silicon Valley demonstrates the importance of flexible specialization and informal networks that Piore and Sabel (1986) borrow from Marshall. In Silicon Valley’s relatively flat, or networked, structure, firms and their employees collaborated, developed alliances, and shared information in informal and formal ways. By contrast, the Route 128 firms had a traditional bureaucratic, or hierarchical, structure that did not afford collaboration. Relatively secretive and self-contained, firms and employees in Route 128 considered the act of sharing information as disloyal. Silicon Valley’s Hewlett-Packard both welcomed new firms into the region and helped incorporate them into larger firm networks. Venture-capitalists were responsible for creating capacity for knowledge and facilitated the transfer of skills between Silicon Valley companies. Route 128 firms, however, fought hard to protect and contain firm knowledge and intellectual property. Part of the contribution of this comparative case study is to show how networked configurations of labor, firms, and thus, skills and knowledge, generated more skills and knowledge at a pace that exceeded that of Route 128.

The firm structure of Silicon Valley was markedly fluid and had capacity to learn from and adapt to adversity. Network-based organizations and startups helped to facilitate the reemergence and revitalization of the region after the crisis of the 1980s. By contrast, Route 128 firms returned to their autarkic form of organization and were unable to keep pace with the innovation of Silicon Valley. Large firms eventually folded, and skilled laborers either moved or made horizontal movements into different industries. Of course, key to the success of Silicon Valley as Saxenian explains, was the participation of universities (including Stanford University, University of California Berkeley, and San Jose State University) in industry and the knowledge spillovers associated with large amounts of R&D. Additionally, firms had strong connections with foreign firms, providing a capacity to reliably outsource and offshore some components of their work. As Saxenian details, this is related to the high number of both immigrants with technology
backgrounds and immigrants with the desire to be entrepreneurs. While the region exhibited a fluid network of labor and the exchange of ideas, it was also connected to both universities and global labor and ideas, which helped solidify the region’s dominance.


The geographic distribution of research activities and the contrasting distribution of private investment and new-firm formation illustrate how both these ingredients need to be combined in order to generate a thriving industry cluster. (Cortright and Mayer 2002, 16)

Both sets of authors have overlapping interests in the biotechnology clusters of the Research Triangle area of North Carolina. While not comparative in the same sense of Saxenian’s work, the biotech industry is equally important to glean insights because of its rate of rapid innovation. Like their high-tech counterparts, biotech firms rely on venture-capital and research to generate and maintain start-up firms. Cortright and Mayer (2001) and Cortright and Mayer (2002) examine the biotech industry and single out the biotech centers in Boston, San Francisco, San Diego, Seattle, and Raleigh-Durham. These centers had a history or presence of research combined with the continued capacity for investment without immediate return on investment. In addition, San Diego, Seattle, and Raleigh-Durham were the largest growing regions. Firms in these regions were able to secure contracts with pharmaceutical firms despite historically narrow odds of regional success.

Different than the high-tech clusters, firms in biotech clusters gain revenue from licensing intellectual property to pharmaceutical companies, or from contracting part of the production process. Regionally, this means the benefit of production happens where the pharmaceutical companies are located, rather than where the knowledge creation
happens in the biotech firms. Of course, the greatest benefit is in locations which have a pharmaceutical presence, biotech clusters, venture-capital, and commercialization capacities.

Feser and Luger (2003) examine how the biotech and education clusters in North Carolina can be used as economic development tools. A key component of their thesis is that cluster analysis is best used as a mode of inquiry rather than as a technical methodology. Abandoning rigid technical methodology allows the necessary flexibility to explore and appreciate the uniqueness of each cluster. Using this type of understanding, Feser and Luger show that the biotechnology cluster of North Carolina is important and deserves economic development. The ability to begin as a mode of inquiry allowed them to uncover the high-intensity of the biotech sector and its large share of national employment. The goal for economic development practitioners should then be to capture as much of the growth as possible because of the region’s already high-intensity. The crossover between the biotech, life sciences, industrial chemical, and environmental industries provides regional advantage that may be missed when observing clusters as a technical methodology.

Nevertheless, because industry clusters, by definition, are not ubiquitous, industry-cluster policy would seem to imply at least acceptance of a potential worsening of regional economic disparities. (Feser 1998, 36)

Feser’s large body of work on clusters not only seeks to understand the North Carolina industry, but also the key dynamics that make these clusters perform. Summarizing the main points of Feser’s work, economic development practice must seek the sources of technological externalities that provide increasing returns, the social and cultural factors as an influence on externalities, the role of economic and geographical proximity as an

There exist clear indicators that successful companies with a trained workforce promote vitality in regions, however, Lowe (2007) best describes some of the potential for true economic development. Lowe suggests that in North Carolina, connecting the two ends of the product pipeline for life sciences – research and manufacturing – allows for a clear workforce development career pathway. Addressing career pathways introduces a new method for sector based economic development strategies. The pathway starts with unskilled young workers working in life-science manufacturing and ends in an advanced degree at a local research center. This describes the product pipeline in reverse, which starts in a research center and ends in manufacturing.

The rapid expansion and increased visibility of these centers and their low relative wages and competitive incentive packages make it increasingly difficult for U.S. regions to compete on cost savings alone. Rather, a region’s ability to retain life science manufacturers will require a deepening of locational advantages other than low taxes, abundant land, and cheap labor. (Lowe 2007, 351)

Few regions have the capability to contain an entire pipeline in the same fashion. This may be unique to knowledge industries, similar in nature to semiconductors, or other forms of cognitive-cultural capitalism.

Of course, the research component maintains its importance, and thus, so does the university-industry linkage. Feser and Landwehr (2006) and later Mayer (2007) examine how the university as a research center can spur the knowledge and skills necessary for the creation of private startups. This is not to diminish the necessary venture or startup capital. In particular, Mayer (2007), Mayer (2005), and Mayer (2012) observes
how in some instances, a university-industry linkage is not necessary, arguing that the university-industry linkages in Silicon Valley were unique. Other regions may rely on large firms to fill the role of generating research or skills, similar to the role that Intel and HP played in Silicon Valley and the role Tektronix played in Portland. Core to this work is the understanding that Silicon Valley’s path of economic development is unique in both place and history, and attempting to emulate Silicon Valley will not yield another world famous cluster. Still, insights from Silicon Valley can be used to spur entrepreneurial dynamics and spinoff processes in other large firms and institutions.

Christopherson and Clark (2007) interrogate the role of innovation within regional production. By surveying the photonics industry of Rochester and the media and production industries of Los Angeles, Christopherson and Clark (2007) assert that regions are not necessarily cohesive units that can capture growth and development. Christopherson and Clark point to first the conflation of regions and the firms that comprise the regions, then to the conflation of innovation (cost-cutting) and production (expansion), as recurring themes throughout their argument. They show that innovation systems do not address power relationships in firm or industry settings. Christopherson and Clark (2007) also find that well functioning labor markets are necessary for the success of innovation and well-functioning networks. A core contribution is to advocate away from investing in corporate “innovation” and instead toward, what sounds like, equity planning.

Chapple et al. (2004) examine high-tech and information technology (IT) centers, looking at the different levels of specialization and concentration among different industry clusters. Using labor classifications to identify industries, this approach is able to identify regions with traditional manufacturing industries that are undergoing transformation. This method highlights the growing interest (at the time) of focusing on human capital. Additionally, their work brings to light the high-tech and IT components of the financial
industries. Markusen, Hall, and Grasmeier (1987) found that industry-linkages and university-linkages were not necessarily requirements for the success of these clusters.

Many of the other sectors identified as I-tech, including finance and insurance, communications, motion pictures, and management and public relations, are not traditionally thought of as high tech, but they support relatively large numbers of IT jobs because of innovations in the way they conduct business. (Chapple et al. 2004, 19)

Building off of that work, Chapple et al. (2004) argue that economic development policies at state and local levels have shifted focus toward on industry-based strategies based on the attraction and retention of home-grown industries and firms. The provided rankings of high-tech and IT centers across the US was hardly surprising, but their work made other key contributions, including hypothesizing the overlap between high-tech and IT industries, identifying the levels of specialization and diversification that each metro had, and signaling for future research on the levels of specialization and diversification on regional economic performance. Chapple et al. (2004) rankings were slightly different in nature than other rankings, spurring Cortright and Mayer (2004) to problematize rankings of high-tech centers as an exercise.

This exchange highlights the importance of combining a thick description of regional production, noted by Feser (1998), with the rigorous quantitative work.

2.5 Emerging Research Questions

Three sets of questions emerge from both theory and practice, begging for new research around software production and regions. Stressing new research is software’s unique relationship between production and output. Software’s output is software, creating a self-fueling cycle of production, distinguishing itself from other forms of production.
Further, the fact that software production relies on human capital for production and innovation – rather than capital machinery and research and development – and the relationship between production and the region, requires a regional examination.

First, around labor and technology, what is the extent of software production activity? This broad requires an understanding of the changing nature of software production and thus an understanding of how software production activity has changed over time. In what ways does capitalist logic manifest itself in software production? How does software production fuel software production and what relationship does software production have to the automation of software workers? What is the relationship between software production activity and larger circuits of production?

Second, around space and place, what are the regionally specific modes of software production? In what ways do industrial districts formations vary? Specifically, what are configurations of firms, regional labor market, institutions, and place and space characteristics. In what ways are they related to Silicon Valley? How do software production activities manifest in terms of regional differences in incomes? Of particular interest to economic development scholars, and directly impacted by a particular mode of production, is the resulting software occupations and the distribution of incomes of software workers.

A third area requires investigation into how the actions of the local actors. Globally, how does the growing numbers of traditionally marginalized groups – women and persons of color – manifest in each region? Locally, in what ways are wages different between regions for these different groups? How are these wage differentials related to the regional mode of software production? In what ways do local actors mediate, either reinforcing or altering, the local mode of software production? In other words, what are the local responses to local income distribution and shifts in occupational structure.

Both software production and regional economies are necessary entry points to under-
stand new capitalist relations. Understanding these new relations thus requires examining how configurations of software production differ across regions, how they impact industry and regional economic development outcomes, and how they are weakened or strengthened by actions of local workers, local organizations, and local firms. These processes offer a glimpse into how the contemporary moment of production differs from other moments of production such as Fordist, post-Fordist, and flexible specialization.
Figure 2.1: Idealized Marshallian industrial district.

○ small local firm

Reproduced from Markusen, DiGiovanna, and Y. S. Lee (1999, 27)

Figure 2.2: Idealized hub-and-spoke industrial district.

○ large locally headquartered firm  ○ small local firm

Reproduced from Markusen, DiGiovanna, and Y. S. Lee (1999, 27)
Figure 2.3: Idealized hub-and-spoke industrial district.

Figure 2.4: Idealized public-sector industrial district.
Table 2.1: Regional characteristics of software industrial districts.

<table>
<thead>
<tr>
<th>Regional Dynamics</th>
<th>Idealized Software Industrial District Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Hub-and-Spoke</td>
</tr>
<tr>
<td></td>
<td>One, or few, vertically integrated &amp; locally headquartered firm(s) outside urban core</td>
</tr>
<tr>
<td></td>
<td>Marshallian</td>
</tr>
<tr>
<td></td>
<td>Large number of clustered small or medium-sized firms</td>
</tr>
<tr>
<td></td>
<td>Satellite-Industrial</td>
</tr>
<tr>
<td></td>
<td>One or more externally headquartered firms, dispersed or clustered</td>
</tr>
<tr>
<td></td>
<td>State-Sponsored</td>
</tr>
<tr>
<td></td>
<td>Government, military, or institutions of higher education, clustered</td>
</tr>
<tr>
<td><strong>Growth &amp; stability</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tied to dominant firm innovation &amp; global exogenous factors</td>
</tr>
<tr>
<td></td>
<td>Growth &amp; resilience potential, tied district innovation</td>
</tr>
<tr>
<td></td>
<td>Dependent on production stability &amp; global exogenous factors</td>
</tr>
<tr>
<td></td>
<td>Steady, dependent on continued government or project funding</td>
</tr>
<tr>
<td><strong>Government</strong></td>
<td>Strong local firm/sector support</td>
</tr>
<tr>
<td></td>
<td>Strong local district support</td>
</tr>
<tr>
<td></td>
<td>External firm incentive support</td>
</tr>
<tr>
<td></td>
<td>Strong Federal/State support</td>
</tr>
<tr>
<td><strong>Migration</strong></td>
<td>Labor in-migration, low out-migration</td>
</tr>
<tr>
<td></td>
<td>Strong labor in-migration, low out-migration</td>
</tr>
<tr>
<td></td>
<td>Strong in/out migration at managerial levels</td>
</tr>
<tr>
<td></td>
<td>In-migration connected to periods of growth</td>
</tr>
<tr>
<td>Idealized Software Industrial District Type</td>
<td>Firm/Production Dynamics</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td>Hub-and-Spoke</td>
</tr>
<tr>
<td>Scales</td>
<td>Moderate-high economies of scale</td>
</tr>
<tr>
<td>Linkages</td>
<td>Ties to large &amp; small firms, locally &amp; globally</td>
</tr>
<tr>
<td>Activity</td>
<td>Design &amp; production</td>
</tr>
<tr>
<td>Risk-sharing</td>
<td>Weak local cooperation among firms, but participation in external industry trade associations</td>
</tr>
<tr>
<td>Services</td>
<td>Local firm-catered finance, technology, &amp; business services</td>
</tr>
<tr>
<td>Labor Dynamics</td>
<td>Hub-and-Spoke</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Labor Market</td>
<td>Internal to firm, hires externally, less flexible</td>
</tr>
<tr>
<td>Turnover</td>
<td>Moderate personnel changes, loyalty to firm</td>
</tr>
<tr>
<td>Profiles</td>
<td>Blue-collar &amp; professional workers</td>
</tr>
<tr>
<td>Intermediaries</td>
<td>Internal, firm, if any</td>
</tr>
</tbody>
</table>
Table 2.3: Table 2.1, Table 2.2, and Table 2.3 adapted from Markusen (1996).
3 Research Design and Methodology

3.1 Introduction

The aim in this chapter is three-fold. First, I introduce research questions that fill pertinent gaps in existing research and have clear implications for equity and economic development policies in cities. Second, I describe the way that my research answers the questions posed, demonstrating the appropriate match between research question, design, and method. Lastly, I describe the data collection and analysis methods. For the technical detail of the GIS-based IPUMS and Census method that I developed for this dissertation, please see the Appendix.

Broadly, this research seeks to contribute to a better understanding of regional software production and advance the conversation on software production oriented-policies and practices that have widespread regional benefit. This research builds on the thorough empirical scholarship of industry-focused regional development scholars (like Castells and Hall 1994; Hall, Markusen, et al. 1983; Harvey 1990; Markusen 1986; Markusen 1996; Markusen, DiGiovanna, and Y. S. Lee 1999; Mayer 2012; Peck 1996; Saxenian 1983; Saxenian Fall 1990) and the in-depth theoretical contributions of industry- and labor-focused political economy scholars (like Massey and Meegan 1978; Amin 1994; J. Graham et al. 1988; Harrison and Bluestone 1985; Noble 1986; Storper and Walker 1989b; Scott 2011a; Wyly 2013) with particular attention to equity, gender, and race/ethnicity in labor markets and industrial production (like Greenbaum 1979; Reskin and Roos 1990;
Colclough and Tolbert 1992; Massey 1995; Chapple 2006; Mayer 2008; Lester and Jacobs 2010). Building on this exhaustive list, I perform a mixed methods comparative case study research design. In doing so, this dissertation will address three sets of questions:

The research is broken into three parts. This first part requires an understanding of how is software produced and how has software production changed over time. The second part requires an understanding of how regions actually produce software and a cataloging of variation in regional software production. This also requires connecting regional modes of software production to equity and economic development through occupation structure and income distribution. Finally, in the third part, I focus on the particular ways in which software work becomes bifurcated and software labor markets exhibits tendencies of dualization and segmentation. The third part also requires understanding how local responses by workers and firms, mediated through the local organization and institutions, respond to the equity and economic development implications of local modes of software production. Broadly, this asks how do is the regional mode of production reified or altered. This final part connects back to the first part. Technological advances in software drive local responses by firms and workers to seek beneficial outcomes from broader changes in software production.

I ask three sets of questions:

1. What is the extent of software production activity in the US? In what ways has software production activity changed in the US since 1970? What is the relationship between software production activity and larger circuits of production?

2. How does software production activity vary between region? How is software produced in Seattle, Washington, Portland, Oregon, and Austin, Texas? How do regionally specific modes of production yield different regional outcomes via occupational structures and income distribution of software workers?
3. To what extent has software work demonstrated tendencies toward labor market dualization and segmentation by gender and race/ethnicity? How are these labor market outcomes mediated by local institutions and practices?

These three sets of questions guide the research methods and subsequently the layout of the dissertation. The first set of questions requires historical research on the evolution of software work and a way to measure software production activity. The second set of questions requires spatialized iteration on the quantitative data from the first set of questions, and combining it with interviews and participant-observation to examine the regional character of software production. I choose three case cities with software agglomerations – Seattle, Washington, Portland, Oregon, and Austin, Texas, described below – to compare to idealized forms of software production and to each other. Lastly, this set of questions requires a way to operationalize the link between software production and occupational structures and income distributions. The last set of questions again requires combining interview data and participant-observation data with a third iteration of quantitative data that breaks down the regional data by gender and race/ethnicity.

Through a mixed methods comparative case study, I will detail the evolution of software production, detail regional modes of software production and identify related regional occupational and wage outcomes, and lastly examine how firms, workers, and institutional actors fortify or alter these modes of production.

3.2 Research Design

These questions require three types of research that cut across the boundaries of the research questions. Specifically, to understand the production of software and how regional modes of software production yield regional economic development outcomes, I will: 1) catalog and characterize the evolution of software production; then 2) extensively
examine regional variation in software production in the US and describe the relationship to types software occupations, the income distribution of software occupations, and the demographic composition of each occupation; and, lastly, 3) intensively examine software production in three US regions through.

I explicitly choose to design this research in what Sayer (2001) refers to as comprising critical research: incorporating extensive, broad based research with intensive, detailed research. I use extensive research questions to measure the extent of relationships and ask about the broad changes in modes of software production. Building from this tradition, I choose an ontological perspective of the quantitative and spatial work of radical positivism (Wyly 2009; Wyly 2011) and critical GIS (O'Sullivan and Manson 2015; Thatcher et al. 2016). This perspective makes explicit claims about the use of quantitative data and research methods in pursuit of equity and social justice.

I use the intensive research questions to understand the how and why the extensive relationships do or do not occur. I seek to understand the relationships change or how they become obdurate. To properly build cases for each questions, I will use multiple informants, multiple sources from various levels, and multiple types of sources. Building off the tradition of Sayer, I situate my qualitative intensive work in the tradition of dialectical research methods for social justice (Agostinone-Wilson 2013). Dialectical research methods help form the types and forms of questions I ask of my interview subjects.

Together, the choices in research questions, method, and data sources seek to identify and dismantle systems of oppression. I use my position as a cis-gendered white male to amplify the stories, and calls to action, of the women, minorities, and persons at various intersections of identity that freely volunteered their time for this research. I cannot fully do them justice in this dissertation. At the end of this dissertation, I reflect on the commonplace notion that software workers are wealthy. But as I uncover, because of the
structure of the labor market and the rapid pace of software development, this benefit does not reach across all software workers. I use the stories in this dissertation to describe the mechanisms, structural and social, that result in this inequity. In chapter 6, I show that the median incomes of women and minorities in the bottom-tier of the software labor market is about $35k a year – less than the amount required for self-sufficiency in each of my case study regions and in stark contrast to the idea that software work produces wealth.

I design this research primarily as case study research across what (Yin 2002, 74) calls five ‘levels.’ At the first and second level, I qualitatively ask questions of firms and workers and quantitatively examine data about the individual region to construct cases within each region. In each set of questions, the third level requires comparison across cases. The fourth level presents all the cases together to learn from the entire study. Each set of questions has small sub questions, but connect to an overarching equity and labor aim. The fifth level presents normative policy questions about regional modes of production and which modes of production might present desirable economic development outcomes. I use three subsections to organize this work, one subsection for each set of questions, which, again, sets the layout for the next three chapters.

Finally, the data sources I detail in the next section cuts across each of the sets of questions. For clarity, I choose to separately describe data collection and data analysis. I do this in the last section of this chapter.

3.3 Research Questions and Method

This section expands on each of the three sets of questions laid out at the beginning of this chapter that serve as the foundation for this research. In what follows, I repeat the research questions, describe the sources of data, and connect the different levels of data to that chapters case. Each subsection leverages the framework set out in the previous
chapter, by “fixing” one of the items in the framework to understand the rest of the items. For example, to understand each software period case, I fix time, industrial district, profit cycle, and capitalist variant. To understand each region’s current industrial district, I fix the software period (fixing time), profit cycle, and capitalist variant. To understand regional labor market dualization and segmentation (part of the industrial district), I fix the software period (fixing time), profit cycle, and capitalist variant. I argue that the profit cycle and capitalist variant are the mechanisms which signal new software periods and changes in the industrial district (and changes in labor market).

Notably, this research will skip Silicon Valley as an intensive place of examination. Much research has already been done in Silicon Valley, and current research by other scholars is currently being conducted there along similar, but sufficiently different, lines. While understanding Silicon Valley can largely be gleaned from the existing literature, research in different places with different modes of software production remain understudied. Again, “as a model for regions elsewhere, then, our research suggests that the Silicon Valley recipe is more complex than has been acknowledged in the literature, less likely to be easily replicated, and less attractive from a social welfare point of view” (Markusen, DiGiovanna, and Y. S. Lee 1999, 292).

To get at the research questions, the research methods in this section are designed to ask about the individual actions that firms and workers take. From these answers, I seek to identify regional firm strategies and worker practices that dictate how firms and workers collaborate, communicate, innovate, profit, and further their own interests. Interviewing firms will require talking to upper/mid-level management within firms (multiple managers in large firms) to identify firm practices, specifically seeking the innovation strategies, regional goals and labor practices. Institutional actors in labor market intermediaries and urban development commissions will interviewed to identify institutional practices that seek to produce or enhance firm or labor networks. Surveying
software workers will require recruiting via tech based networks like Hacker News, Twitter and Reddit and to identify labor practices; specifically seeking how workers innovate, why who their network is, their labor mobility (spatial and firm), and how they use their network. Both firm-level and worker-level data will be analyzed to confirm, refute or identify regional firm and labor practices.

Through this analysis, I contend I will get at the motives, both capital and non-capital, that drive these decisions, fortifying or altering the configurations that constitute regional modes of production and thus impact economic development outcomes. Further, firms and workers seek the most beneficial mix of capital, labor, space and place, this questions attempts to understand what configurations yield the best outcomes for both firms, workers, industry and region. With this explicit link, this phase will contribute to an understanding of how firm strategies and worker practices fortify or alter regional development outcomes through regionally specific modes of production.

3.3.1 Labor/technology: Extent and restructuring of software production

1. What is the extent of software production activity in the US?

2. In what ways has software production activity changed in the US since 1970?

3. What is the relationship between software production activity and larger circuits of production?

With this set of questions, I seek to broadly understand software production and the extent of its change over time.

(occasionally for brevity: 2009) and 2011-2015 (occasionally for brevity: 2015). This allows me to capture change over 10-year periods to 2000, then approximately 7-year periods after that.

Using this data, I identify software occupations, and measure the extent of software workers at each point. I also measure the integration of software workers into US industries by building off measures of industrial specialization. I describe this method in detail in Chapter 4. The idea is to determine the degree to which software workers concentrate in any one industry. The results of this inform the rest of this research, confirming preliminary interview research which suggested that Portland software workers come from many industries. These finding reinforce the idea that software workers are diffuse across industries and thus measure software production activity requires measure software labor, not using an industry-based approach.

Thus I measure the extent of software production activity as the extent of software labor at any point in time, and in the next section, in any given metro region. With measures of software production activity at six point in time (1970, 1980, 1990, 2000, 2005-2009, 2011-2015) and a measure of the degree of integration into US industries, I describe how the extent of software production has changed over time. Matched with historical research on changes in software production and software commodities, and peppered with the unexpected accounts of software labor from my interview subjects, I describe the changing nature of software production in the post-WW2 period. Historical research allows me to identify key firms and organizations that shaped the extent of software production today.

As I describe the changing nature of software production, I connect software production periods to product cycle periods (each containing 5 profit cycles). I note that during each commodity period, the software commodity form has a new product cycles indicated new forms of labor, capital, and spatial relationships. In each transition period, software
undergoes restructuring that reconfigure the sets of dominant actors in software creation and use. I draw comparison to the capitalist variants literature. Each of software’s periods match particular sets of capitalist variant characteristics. Capitalist variants intend to describe economy-wide characteristics, yet, software’s microcosm of production mimics these relationships. I note that the product cycle of each software period is the driving mechanism for the capitalist variant to change.

3.3.2 Space: Regional variation and regional production

1. How does software production activity vary between region?

2. What are the industrial district formations of software production in Seattle, Washington, Portland, Oregon, and Austin, Texas?

3. What are the regional outcomes, via occupational structures and income distribution of software workers, of regionally specific modes of software production?

With this set of questions, I seek to understand how software production has regionally changed over time, the local nature of software production, and how it has evolved into its current form in Seattle, Portland, and Austin. I seek to understand how software is locally produced through regionally specific modes of production and the of income distributions and occupational structures connected to regional production. I build on the idea the modes of production have specific industrial districts – requiring the investigation of each of my case region’s industrial district.

Most regions have software agglomerations that track with population and software is likely produced uniformly. First, I analyzed US metro regions over 1 million people in population between 1970 and 2015 to examine the regional variation in software production. I measure regional software agglomerations, by measuring the quantity of software workers and the share of the labor force that is comprised of software workers.
I seek to track regional changes in relative rankings between 1970 and 2015 to better understand the regional impacts of the restructuring of software periods addressed in the previous set of questions.

Then I seek to identify regional modes of software production. The literature suggests that Silicon Valley, Seattle, Austin, and Portland may exhibit regionally specific modes of software production that can further analyzed by uncovering each regions software industrial district. I then use interviews, historical research, and participant-observation to identify the characteristics of each region as outlined in Table 2.1, Table 2.2, and Table 2.3. Through this data I identify key firms, organizations, groups, and institutions that shape software production in each region. I then use this to shape future interviews and participant-observation, ultimately triangulating and uncovering each regions’ configurations of firm strategies, worker practices, regional configurations, and their relationships. I further seek to identify specific technological characteristics specific to software industrial districts. Again through interviews, historical research, and participant-observation, I identify regional variation in software use, forms of automation, and the role of technology in innovation. I document these characteristics in Table 5.3. I then compare each regions industrial district to the ideal types of industrial districts identified by Markusen, DiGiovanna, and Y. S. Lee (1999) and type formulation suggested by Brenner (2013).

Additionally, four hypothesized regional modes of production may present, expanded in the next section.

- Silicon Valley is the archetypal software region, which the literature suggests was once prototypically Marshallian. It is a unique global hub of semi-conductor and software industries. Examining Silicon Valley expands our knowledge base about the archetype, but for reasons outline above, I skip engaging directly with Silicon Valley and instead choose to understand its connections to the three case study cities
Seattle is a stereotype of the hub-and-spoke industrial district. Seattle’s primary software firms are Microsoft and Amazon. Seattle has a particular spatial and industrial history which has produced its current industrial relationship.

Austin is a stereotype of satellite-platform industrial district. Again, Austin has a particular spatial and industrial history which has produced this type of industrial relationship. Austin’s primary software establishments are branch plants of Apple, Google, Microsoft, National Instruments Delphi, etc.

Portland is a prototype of a Marshallian district that is characteristic of production under cognitive-cultural capitalism. First, this research will compare each region comparatively rather than each of the above regions to just Portland. Second, this research acknowledges that other regions may not develop in the same trajectory that Portland has developed or is developing (ie, there is no single development path). This research avoids the prototype-trap not suggest that the prototype is the ideal-type for regional economic development outcomes.

I then constructed industrial district typologies for each region based by comparing the collected data to the characteristics of industrial districts documented above. With this part of the dissertation, I identify each of my case city’s regional modes of production and how they map onto theorizations of industry typologies and regional production (industrial districts) as part of broader industry restructuring. For each region, I also describe the regions relationship to Silicon Valley to situate the region in larger networks of production. Then I examine the industry composition of software workers to determine the areas of employment for software workers and the types of companies that might employ them.
To address the final questions, I examine regional outcomes measured by distribution of incomes and types of software occupations. Specifically, I look for software differences in equity by looking at the shape of income curves and compare them to the expected outcomes based on the industrial district. I also compare the occupational structure – the amount of different jobs and their skill-level – with expectations we might have for the industrial district. Here we see how regions reflect their industries and labor-markets, and the processes for which regions are “produced” through differing regionally specific configurations of production. I contend that these both software worker based outcomes onto distinct regional modes of production and yield significant differences in income distribution and occupation structures. Examining occupational change over time requires identifying how occupations have changed over time. I thus document the changing occupation codes, which reflect the underlying work of laborer. I combine these changes with information from the literature and the accounts of software workers to identify moments of labor market bifurcation. This, along with income data, enables me to categorize occupations based on their labor market tier, and understand the historical tendencies of labor market dualization and segmented.

This chapter integrates space into the discussion and uses industrial districts to integrate space into the framework. As I connected software periods, product cycles, and capitalist variants, I use the relationship between the profit cycles of product cycles to connect to industrial districts. The needs of firms are reflecting in their locational decisions as they progress through profit cycles. These needs are expressed in the forms of industrial districts. Since software spans multiple industries, the software commodity product cycles becomes most relevant, rather than the profit cycles of the industries that use/purchase/alter software commodities. Thus, the industrial districts reflect the relationships of software production, rather than industry relationships. For software, industrial districts reflect the form of the software commodity and the capital relationships
from capitalist variants.

### 3.3.3 Wage structures: occupations and their income distribution

1. To what extent has software work demonstrated tendencies toward labor market dualization and segmentation by gender and race/ethnicity?

2. How are these labor market outcomes mediated by local institutions and practices?

The last set of questions seeks to understand outcomes of the structural changes of software production (technology/labor) with the regionally specific nature of software production (space). I combine my interviews, historical research, and participant-observation with a third iteration of the IPUMS data, this time adding race/ethnicity and gender. With the IPUMS data, I document the regional occupational, then tiers of the labor market, breakdown of software work. I group the occupations, based on income and function, into occupational groups and tiers, comparing national statistics across tiers based on gender, race/ethnicity.

I then examine the local variation in software work across tiers for each region. I examine the variations in incomes across race/ethnicity and gender, referring to the constructed industrial district (from the previous set of questions), interviews, historical research, and participant-observation to explain unexpected variations and similarities.

To understand how local institutions mediate or mitigate structural labor market forces, I used the transcriptions from my interviews to identify actions that workers take which concretize, challenge, or change the ways that deskilling, segmentation, racism, or sexism impact local labor. I then group these responses by their action and by the mediation. I draw mostly from my interviews with women and minorities as these were the most informative on the topic. Sexism was the primary focus of my interviewees, even those women of color. This speaks to the literature that asserts “... occupational
sex segregation has been more resistant to change than race segregation” (Reskin and Roos 1990, 6).

### 3.3.4 Mixed-methods research

One core methodological arguments is that the contemporary landscape of software production – and the changing character of technology under capitalism itself – requires the tracking extent of national and regional change over time alongside an intensive investigation of regionally specific software production. Each research question thus synthesizes both quantitative and qualitative research data to construct detailed responses to each of the questions posed. Extensively, I measure historical, regional, then occupational changes. Intensively, I measure the changing nature of software, relationships between firms and workers, and the forces underlying deskilling and segmentation. I elaborate on this in Table 3.1.

### 3.4 Case Selection

#### 3.4.1 Seattle, Portland, and Austin

One key variable in this research is the role of small firms in labor markets. Smaller-firms may not be as equitable in pay for women or persons of color, nor have the same regional economic impact. Each of these scenarios might dictate differences in the types of software occupations and hierarchies (high-level vs lower-level) present both within the local industry and the broader region, the tendency toward types of labor networks, and the levels of worker incomes. Further, the clustering of jobs or firms might also enable different types of networks or cultures; some regions with smaller-firms may have laborers that cooperate to maintain regional skill levels and compete with software production in other regions. This largely depends on whether firms, especially large firms, control the
Table 3.1: Synthesizing mixed-methods research.

<table>
<thead>
<tr>
<th>RQs</th>
<th>Quantitative</th>
<th>Qualitative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Extent/change of software production</td>
<td>Number of software laborers in each software period; software workers by</td>
<td>Categorizations of software languages; software worker accounts across times;</td>
</tr>
<tr>
<td></td>
<td>different industry in each software period; measures of software worker</td>
<td>industry brochures and magazines (historical and interview-based)</td>
</tr>
<tr>
<td></td>
<td>diffusions in each software period (PUMS data)</td>
<td></td>
</tr>
<tr>
<td>2. Regional modes of software production,</td>
<td>Regional subsets of software laborers in each software period and their</td>
<td>Firm strategies and labor practices across regions to construct industrial</td>
</tr>
<tr>
<td>industrial districts</td>
<td>industrial composition; tiers and classifications of the various occupations</td>
<td>districts; regional characterization of software production, values,</td>
</tr>
<tr>
<td></td>
<td>in the most recent period; comparison of median incomes across regions</td>
<td>interactions; software-specific variations in idealized industrial districts</td>
</tr>
<tr>
<td></td>
<td>(PUMS data)</td>
<td>(participant-observation and interview-based)</td>
</tr>
<tr>
<td>3. Labor market segmentation, local mediation</td>
<td>Historical occupational bifurcation tracking; national and regional income</td>
<td>Local and regionally specific responses to labor market segmentation,</td>
</tr>
<tr>
<td></td>
<td>distribution by gender and race/ethnicity; classification of software</td>
<td>dualization; relationship between industrial district and local labor power;</td>
</tr>
<tr>
<td></td>
<td>occupations by labor market tier (PUMS data and Bureau of Labor Statistics</td>
<td>emergence, and type, of local labor institutions; priorities of regional</td>
</tr>
<tr>
<td></td>
<td>occupational code tracking)</td>
<td>software workers; resistance to firm strategies (participant-observation and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>interview-based)</td>
</tr>
</tbody>
</table>
network of production consisting of the control the local labor market, local contracting firms, and other potential non-local collaborators.

Labor cultures of openness and combined with spatial proximity may be linked to strong production networks, mimicking the same economic development benefits that make locally headquartered large firms in the economic development literature. Clustering of software jobs and firms in the urban core might heighten the pressures of gentrification while regions with primarily suburban jobs and firms might be linked driving new home builds – both affecting the locations and levels of concentrated poverty and racial/ethnic segregation and complicated further by the presence of large locally headquartered firms that may positively contribute to a broader base of employment.

The industry specialization or diversification of both firms and workers suggest different growth and resilience trajectories. Specialization may provide higher incomes and faster growth, but diversification may provide better protection from macroeconomic shock. Whether these factors are important for firms or workers is part of this research.

I choose Seattle, Portland, and Austin because they are second-tier regions that compete with Silicon Valley and contribute to the landscape of software production generally. Silicon Valley, itself, may not provide useful insights into the software production and regional modes of production because of its archetypal status. Each case region connects to Silicon Valley through firms and workers while exhibiting variations on the key variables identified above: firm sizes, firm interconnectivity, firm clustering, firm strategies, labor connectivity, labor diversification, maturity of software production, and historical development. Thus, each region provides insights that may help us understand software more generally, regional modes of software production within each region, and draw comparison to regional modes of software production in other second-tier cities.
3.4.2 Hypothesized industrial district formations

Silicon Valley, as a once Marshallian high-tech manufacturing district that now serves as the archetypal software region, will likely have produced a hyper-bifurcated labor market as firms attempt to routinize software work. The software agglomeration of Silicon Valley attracts both high-skilled labor and unskilled labor as firms and laborers attempt to share in the profit; however, the continual pressure to routinize software work or automate software development through high-level language adoption results in active deskilling of software labors. As firms continue to grow and advance through profit cycles, firm consolidation results in higher-levels of hierarchical forms of firm organization. Further, this bifurcation might lead to a polarization within the industry and region, with the lowest levels of software work being the first to be cut during the financial crisis. And yet, held in dialectical tension, the access to high-levels of venture-capital and the potential for firm-consolidation (being bought-out) means that small startup firms are attracted to can present laborers the opportunity to generate wealth very quickly. Thus, we would see very large enterprise corporations at the outskirts of the city and large numbers of smaller corporations in the central city where laborer interaction is necessary – both locations containing a mixture of high-level and lower-level software jobs.

Seattle, as the traditional hub-and-spoke district, may have a concentration of high-level software workers as a result of the firm strategy to outsource and offshore low-level software work – different than the firm strategies in Silicon Valley. This might produce high-paying occupations within the software industry and for software workers, but present barriers for software workers to gain access to these jobs because of high-barriers of firm-entry. The higher quantity of both large Seattle firms and large Silicon Valley suggests that that firms in these regions tend to innovate by acquisition of smaller firms. These higher skilled jobs are likely to be outside the city.
Portland may provide a mix of high-level jobs and low-level jobs because of the large number of medium sized firms and an absence of venture capital compared to Silicon Valley, Austin and Seattle. However, the industry growth may be muted without access to the venture capital of Silicon Valley or Seattle to spur the growth of these firms. Firms have been notorious in Portland for refusing venture capital investment as a firm strategy. In fact, numerous articles in the Oregonian highlight this attitude, highlighting that attitude that smaller Portland firms must collaborate to attract high-skilled laborers and avoid the control that comes with accepting venture capital. While Portland firms may resist outside capital, non-Portland firms like eBay and ESRI have recently been opening locations within Portland seeking to tap the relatively high-skilled and inexpensive labor within the city. Lastly, Portland innovation may not happen within the firm but between laborers at various cooperative gatherings, suggesting higher rates of skill-trading between labors. The once endogenous nature necessary to ‘skill’ Portland labor might be changing, as new openings from secondary locations of non-Portland firms seek to tap the innovation cycles of the city. Portland may be like Silicon Valley with a spatially dichotomous software jobs, but have higher-level jobs within the city and lower-level routinized jobs outside the city.

Austin may have the exact opposite problem as a typical satellite-district, producing numerous low-level jobs with fewer career ladder opportunities to grow into higher-level jobs. And while levels of software workers and software related industries may be high, these types of jobs would be skewed to lower-end jobs, software testing and software support firms rather than development. Austin, one a scene for high-tech hardware with Dell, National Instruments and various memory manufacturing nearby, has had difficulty in a once prosperous area. Recently, Apple a software firm that arguably distribute software through its hardware, recently opened up a 4,000 job location in Austin to serve support functions for its products. Little innovation is necessary for these support types
operations and there may be no spatial concentration of these types of low-level software jobs.

3.5 Data Collection and Data Sources

3.5.1 IPUMS USA, Census 1970 to 2015

The Integrated Public Use Microdata Series USA (IPUMS USA) data provides scholars with the ability to perform cross tabulations on US census data (Ruggles, Alexander, et al. 2010). This differs from the count data provided directly from US Census Bureau websites such as DataFerrett or American FactFinder. For example, using IPUMS USA, scholars may determine the number people in the occupation Computer Programmer that also identify as Hispanic, or any other combination of race and ethnicity, and median income. Using data provided by the US Census Bureau only allows for users to determine the number of people in the occupation Computer Programmer or the number of people that identify as Hispanic, not both.

For urban planning, urban studies, geography, and sociology scholars, especially those interested in equity and social justice, the IPUMS USA is a powerful data source. However, scholars do not frequently use the IPUMS because the default geography of their data and the changing nature of the geography. The IPUMS USA uses a geography called the Public Use Microdata Area (PUMA). PUMAs are the smallest geographies used by IPUMS USA and consist of 100,000 to 200,000 residents. PUMAs can intersect geographies typically used by academics, such as counties or metropolitan areas (Core Based Statistical Areas, or CBSAs). Further, between census years, PUMA boundaries can change drastically. These characteristics make using the IPUMS USA difficult to use and disseminate research that uses the metropolitan area as the unit of analysis and even more difficult when performing change-over-time analysis of metropolitan areas.
For this research, I use the Census 1970 1% metro form 2 sample, Census 1980 5% state sample, Census 1990 5% sample, Census 2000 5% sample, ACS 2005-2009 5-year sample, and the ACS 2011-2015 5-year sample. This provides 6 comparable snapshots of data. In Appendix A, I describe how I operationalize and spatialize this data.

The key variables for this research are:

- **OCC** describes the occupation code according to the current years occupation classification. This occupation classification changes over time, and I document these changes as part of my research.

- **OCC2010** uses the 2010 occupation codes and reverse codes occupational status for each historical sample. This provides the easiest way consistently put boundaries on what counts as software labor through time. I use the occupation codes OCC2010 provides occupation classification and historically codes 110, 1000, 1010, 1020, 1050, 1060, 1100, 1200, 1220 which I describe in Appendix B.

- **INCWAGE** is the income from wages for workers. Incomes over a certain percent are top-coded, making means difficult to compute. Instead, I compute medians for incomes where possible.

- **IND1990** provides 1990 industry classification and is reverse/forward coded for the entire sample of this dissertation. It is the most recent industry classification that covers all samples. I use this is any industry classification that requires going back to 1970. Importantly, because of changes in reporting, I exclude software occupations in Public Administration and Active Duty Military are not used due to historical comparability, which are codes 900 and over. One might expect this to affect Washington DC, but it did not appear to have an impact on relative rankings.
• **INDNAICS** is a modern industry classification used when describing 2011-2015 data.

• **SEX** provides the sex of recipient which I refer to as gender.

• **RACE** and **ETHNICITY** provide data on race and ethnicity. I combine these to construct groups of white non-Hispanic (NH), Black NH, Asian NH, Native American/Alaska Native NH, Other NH, and Hispanic/Latino. As I explain in a second, due to issues of standard and small sample sizes for minorities, I group these further into white NH, Asian NH, and all other.

One issues with the IPUMS USA is large standard error when performing multiple cross tabulations. For example, the low inclusion of Black and Hispanic/Latino in some of my regions means that getting detailed information about income and occupation run into significant challenges of data reliability. This would also be difficult if I were to do intersections of race/ethnicity by gender for software workers. For this reason, I avoid going another variable deep in the research (currently occupation and one of gender or race/ethnicity). This is an issues to be aware of for future research.

### 3.5.2 Interviews

I conducted 33 semi-structured interviews between August 2015 and November 2016 in Portland, Austin, and Seattle. To initiate the interview process in each region, I used the internet to find software workers that were active in each regions’ local Python community by browsing Python meetup pages and user-group postings. This provided an even entrance into each community. I identified 5 people in each region using this method. I received replies from 12 software workers through whichever means the individual seemed most active: LinkedIn, Twitter, GitHub, or email if none of these. In each region, 2 of the 5 individuals were visibly women. I explained my interest in interviewing
software workers, explained the research project, provided the IRB consent form, and asked if I could interview them for 45 minutes. I recommend meeting at a coffee shop at a time that suited them, but offered that we could meet at another location of their choosing, or alternatively a video call. I separately contacted the economic development agencies in each of the regions. I conducted interviews of tech-oriented economic development practitioners in Portland and Austin. I did not receive a response from the Seattle or Redmond economic development agencies.

I performed 5 practice interviews in Portland that were not included here to test out the interview questions. To gain the fullest understanding of each region with limited interviews, I created a set of guidelines to steer my research. I sought to perform 8 interviews in each region. I sought to have at least one-third of the interview subjects be women (8), reaching at least 2 in each region. I also sought to interview visible minorities in each region, however, this proved much more difficult than I expected. I sought to interview at least 2 managers from each region. After initial interviews in each region, I used a snowball sampling technique to interview other software workers. During interviews I asked what kinds of places characterized software work in the region (firms, conferences, institutions, meetups), then asked if they could connect me to software workers they know at those places. I asked to speak to former managers, women, and minorities. I was initially concerned that by starting with the Python community, I would miss important actors in each region, however, this proved not to be an issue where there were sufficiently large actors, even if they were somewhat disconnected (as in Austin, which I explain later). In Seattle, I was able to speak to workers at Microsoft and Amazon. In Austin, I was able to uncover workers at the University of Texas and the State of Texas.

My interviews exceeding the interview subject guidelines I set. Below is a summary of my interviews.
• 33 interviews of software workers – 13 in Austin, 11 in Portland and 9 in Seattle.

• 27 recorded and transcribed interviews (6 asked not to be recorded).

• 13 women software workers, at least 2 per region, 39% total.

• 1 transgender software worker.

• 7 non-white software workers (1 Black, 3 Hispanic, 3 Asian).

• 5 non-white women software workers.

• Austin (13): 6 with experience in local public-sector, 5 with experience at local SME, 3 with experience at a locally headquartered large firm, 2 with experience local branch office of large firm. 3 currently self-employed. 4 with experience in Seattle, Portland, or Silicon Valley. 3 managers. 2 “district champions.”

• Portland (11): All software workers had experience at a local SME at some point, 3 with experience at local branch office of a large firm, 2 with experience at a locally headquartered large firm. 2 currently self-employed. 4 with experience in Seattle, Austin, or Silicon Valley. 4 managers. 2 “district champions.”

• Seattle (9): 8 with experience at a locally headquartered large firm, 2 with experience at a local SME, 1 with experience at local branch office of a large firm. 6 with experience in Austin, Portland, or Silicon Valley. 6 software workers moved to Seattle region to work at a large locally headquartered large firm. 4 managers. 1 “corporate champion.” All 4 in-person Seattle interviews conducted at Starbucks by interviewee request.

• Firms: Across regions, a sample of the firms that software workers in this dissertation include: Microsoft, Amazon, IBM, Disney, Mozilla, Netflix, Hulu, Open Sourcery, New Relic, Puppet, Heroku, University of Texas, State of Texas.
Groups: Across regions, a non-exhaustive list of the local groups and events that workers attended: PyLadies, PyMoms, AustinFeminist, DjangoGirls, local Python Users Group, Latina Hackers, Girl Develop IT, Gem Austin, Girl Start, Women Who Code, Girl Connect, Texas Girls Collective, Code Dojo, Girls Who Code, Ruby Users Group, local Linux Users Group, SXSW, OSBridge, OSCon, PyDX, atxrails, local hackathon, Microsoft Build, Women In Tech, Python Flying Circus, Safety First, pdxrlang, Perl Mongers, and Puget Sound Programming Python. There were others that software workers attended in the past that had stopped attending.

I asked questions in six broad sets of categories.

1. History of entrance into software work, move to current region

2. Daily commute, typical day, typical sets of software/languages, typical interactions with managers/subordinates

3. Connections to software workers outside group/firm

4. Connections to firms and software workers at other firms

5. Regional character, local events, connections to software workers in the region

6. Skillbuilding, ideal set of skills, ideal candidate

Additionally, as a participant-observer, I attended Python events in each region, volunteered at conferences in Portland and Austin, and attended hackathons or other hacking events in each region. In total, I attended 9 events, 2 in Portland and Seattle, 3 in Austin. As a method of gaining access to these conference, my volunteer status meant that I performed tasks like checking people-in, patrolled hallways as an “Incident Response Monitor,” or assisted in stuffing conference bags. These events allowed me to corroborate the answers of workers, their characterizations of the regions, and to ensure that I did
not miss any significant firms or institutions. It also allowed me to identify potential interview subjects, although I only interviewed 1 person because of attendance. I kept a field journal of these events and used this extensively when describing each region’s industrial district.

While the interviewees aren’t necessarily demographically representative of the software workers in each region, the breadth of each experiences and knowledge of the software workers suggests that triangulating characteristics of each region from the interview subjects would be reliable. Yet, reaching black software developers, or Hispanic developers in Seattle, proved to be the most challenging part of this research. Further, reaching black women proved near impossible outside of a discussion at conference — and she did not know other black women in the region that were doing software work. With such low representation of women and minorities in software work, these challenges are not necessarily surprising, but future research could start by trying to identify groups like Austin’s Latina Hackers.

Members of the LGBTQQ communities are also underrepresented in this research and have arguably more difficulty in the tech scene than other marginalized groups. I did not ask interview subjects of their sex or sexual orientation, but one interview subject openly identified as transgender. This individual faced significant challenges in the workplace which I cannot capture in this dissertation. A second pass at data collection could focus explicitly on members of the LGBTQQ community.

Lastly, as a cis-gendered white male asking electronically to speak potential interview subjects that were non-white, women, or non-white women, I was confronted with the question: “what’s in it for me?” This turned into one of the longest interviews I performed, but it also makes clear that my perceived identity impacted the initial reaction of interview candidates and most-likely their responses. Unsurprisingly, my most successful method of interviewing women and minority women was through referral. Still, because of
my positionality, my interview subjects likely, knowingly or unknowingly, altered their responses and method of interaction. While I seek to use my privilege to elevate the voices of those marginalized, the different responses that I may receive from interview subjects and my analysis of the quantitative data may threaten the validity of my collected data and research analysis respectively.

I transcribed all 27 of the recorded interviews using MaxQDA. I created a coding scheme to based deskilling, issues of race/ethnicity, issues of gender, and left hand categories of the industrial district typology tables (2.1, Table 2.2, Table 2.3, and 5.3). I used these to triangulate the industrial district typologies, document the racialization and gendering of software work, and then later address how structural forces are mediated locally through different institutions (use left hand codes from the industrial district typology).
The link between a decline in the economy-wide rate of profit and structural realignments within each industrial sector lies with the stresses created by heightened competition for funds. Innovative firms which are quick to introduce new technology, labor practices or organizational stratagems can cope with competitive stress successfully.

J. Graham, Gibson, Horvath, and Shakow 1988, 478

Software is a funny thing. You can’t touch it, but you can certainly pay for it. It weighs nothing, but takes dozens or even hundreds of people to make. And sometimes it just doesn’t work out.

Jensen 2016, in PCMAG

4.1 Introduction

Software production and its relationship to regional economies remains misunderstood or misidentified in the literature, contributing to different future imaginations and an inability for scholars and practitioners to leverage software production for economic development. Economic development research has long believed in the power of technological-related productive activities to produce good jobs (Markusen 1986; Morgan and Sayer 1988; Cortright and Mayer 2001; Saxenian 1983). More recent research questions this takeaway, arguing that software, as part of a suite of technology-related productive activities, does not alone produce broad-based regional wealth, but it does produce wealth (N.
Lee and Rodríguez-Pose 2016). A broader argument of this dissertation is that software production activity does have the capacity to produce broad-based wealth but with certain conditions. This chapter asks three questions:

1. What is the extent of software production activity in the US?
2. In what ways has software production activity changed in the US since 1970?
3. What is the relationship between software production activity and larger circuits of production?

To investigate these questions, and following the epigraph at the beginning of this chapter from J. Graham, Gibson, Horvath, and Shakow (1988), this chapter will trace the restructuring of software production activities focusing on changes in technologies, labor practices, and organizational strategies across US industries. Through data from the US Census (1970, 1980, 1990, and 2000) and American Community Survey (2005–2009 and 2011–2015) of software workers six time periods, this chapter tracks the extent and nature of software production. Primary interviews, combined with secondary historical accounts of software production, explain the conditions and circumstances of these the four software periods, describing the specific mechanisms of change on the “shop” floor and in the firm.

Directly addressing this chapter’s questions, this chapter lays out three arguments:

1. First, to understand software production activity, scholars must measure the extent of software labor. Specifically, scholars must reject the ease of identifying one or more software “industries” or sectors to understand software production activity because workers produce software across industries as a integral component of capitalism.

1“strategems” in the epigraph
2. Second, a fourth period of software and software production activity started in the mid- to late-2000s. This fourth period, distinct from the mass-market and desktop computing of the previous period, draws on the connected nature personal devices, web/mobile apps, and cloud computing as part of a “cloud and mobile” software period.

3. Third, software production’s periods – and the restructuring between periods – falls along three interrelated concepts: variations of capitalist relations, firm/industrial profit cycles, and the spatial and social formations of industrial districts. Specifically, software production’s four periods have a relationship to dominant mode of production. Each of these periods has specific profit cycles and industrial district formations. I argue that the transition of software production along these capitalist variants, profit cycles, and industrial districts brings new insights into the other forms of capital production. Further, the form and changes within software production activities may be prototypical for other types of production and its respective restructuring, geographies, and divisions of labor.

This chapter is organized in four software periods.

4.2 Software Periods and Software Production

In Table 4.1, I revise the software periods introduced by Campbell-Kelly (2004) and Campbell-Kelly and Garcia-Swartz (2015), updating them with new insights from data past 2010. I elaborate the description of each period to include the dominant computing platforms, the types of software produced, and the forms of software production. As I will show, the connected nature of fourth period is an important and distinguishing factor from the previous period. In the mass-market software period (third), software did connect over the internet, but in the cloud and mobile period (fourth), software runs
on and requires connection. For example, Google Docs and the new forms of Microsoft Office require both client and server to run. Software is also continuously updated, or licensed, rather than purchased. Users of ESRI ArcMap or IBM SPSS know that licenses are typically year long, and even these applications are moving to cloud format like Google Docs and Microsoft Office. Continuous updates also allow for new competitors to create simple proof-of-concepts, and then expand functionality as the user base grows or as user requirements change. Software production in this period also directly engages with clients and users, which, for open source, means that clients and users can contribute back to the production of the software.

In Table 4.2, I extend the work of J. Graham et al. (1988) by adding a fourth, cognitive-cultural capitalist variant based on the work of Scott (2011b) and Wyly (2013). I elaborate on the concept of cognitive-cultural capitalism by introducing networked machine-learning as a form of automation and relationship between capital and labor (eg software used to write software). I introduce the idea of computational flexibility as the labor process where software algorithms determine laborer shifts (eg fast-food restaurants) as well as determine contracting (eg Uber selecting drivers for clients). Capital continues to concentrate, yet, digital fixed-capital, like servers, are increasingly decentralized or moved to the periphery (eg Google’s data center in The Dalles, Oregon). Digital capital, as opposed to productive capital, is a circuit where commodities are also the means of production, owned by the commodity producer, to collect the labor of buyers (eg writing reviews, social media, or location tracking turned into commodities). I use “digital capital” to describe this, combing work on Marxian internet studies by Fuchs (2012) and Rigi and Prey (2015) and work on platform capitalism by Srnicek (2016). This is, at best, rough and somewhat hypothetical, yet examples are readily available outside of software production. Within software, these relationships and conditions are already evident. Understanding and explaining this elaboration of cognitive-cultural capitalism requires
future work and I make no claims on fully explaining it here. For now, and for software production, I operationalize cognitive-cultural capitalism to explain the use of software to write software, the use of software in an already flexible labor market, and the emergence of large, external data centers.

In Figure 4.1, I introduce a figure to describe a framework for analyzing software production, arguing that software’s first period was under conditions of competitive capitalism, that software’s second period was under conditions of monopoly capitalism, and so on. I apply work on industrial districts by Markusen, DiGiovanna, and Y. S. Lee (1999) to argue that the during each of the software periods, certain types of industrial districts flourish. Each chapter contributes to this figure and its argument. For this chapter, the relationship between capitalist variants and software periods is most important, and I elaborate in-depth on industrial districts in Chapter 5.

4.3 1950s and 1960s: Contracting, Increased Competition, and Punch Cards

4.3.1 Military spending, universities, and entrepreneurs

Modern constructions of the history of Silicon Valley claim that military spending, universities, and entrepreneurs play important roles in the success of the region (Saxenian 1996). These ingredients find similar histories in software and computing more generally. Origin stories of software and computing vary, but the computing projects funded by the US government required significant advancement in technology, required the research knowledge at universities, and required programming at a previously unprecedented scale. I will focus on two of the well-known computers of the 1940s that comprised the largest computing projects of the time. The ENIAC computer, the Electronic Numerical Integrator and Compiler, was secretly developed at the University of Pennsylvania during World War 2. The Mark I computer (also the Automatic Sequence Controlled
Table 4.1: Periods of software.

<table>
<thead>
<tr>
<th>Software Periods</th>
<th>Contracting</th>
<th>Corporate</th>
<th>Mass-market</th>
<th>Cloud &amp; mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computing platform</strong></td>
<td>Large computer</td>
<td>Mainframe &amp; terminals</td>
<td>Desktop computer</td>
<td>Cloud &amp; interconnected devices</td>
</tr>
<tr>
<td><strong>Types of software</strong></td>
<td>One-off projects</td>
<td>Specialized software on specialized computer, fixed / static programs</td>
<td>Operating system, desktop software, flexible programs</td>
<td>Mobile / web apps, server synchronization</td>
</tr>
<tr>
<td><strong>Software production</strong></td>
<td>Software services, specialized</td>
<td>Software suppliers</td>
<td>Software publishers, physically distributed, major releases, silo development</td>
<td>Networked, continuous updates, community contributions, replaceable / interoperable</td>
</tr>
</tbody>
</table>

Adapted and extended from Campbell-Kelly (2004, Figure 1.1).
Table 4.2: Capitalist variants.

<table>
<thead>
<tr>
<th>Structural relations of exploitation (capital-labor)</th>
<th>Structural element</th>
<th>Capitalism in general</th>
<th>Historically specific capitalist variant</th>
<th>Structural innovations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions of exploitation (capital-labor)</td>
<td>Technology</td>
<td>Competitive</td>
<td>Monoploy</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Machinofacture</td>
<td>Semi-automation / mass-production</td>
<td>Full automation / robotics</td>
<td>Machine-learning / networked</td>
</tr>
<tr>
<td>Labor process</td>
<td>Modern industry</td>
<td>Fordism / scientific management</td>
<td>Flexible / post-Fordist</td>
<td>Computational / flexible</td>
</tr>
<tr>
<td>Reserve army</td>
<td>Redundant labor</td>
<td>Segmented labor</td>
<td>Peripheralized labor</td>
<td>Global periphery, laborer atomized</td>
</tr>
<tr>
<td>Conditions of competition (capital-capital)</td>
<td>Centralization</td>
<td>Regional horizontal</td>
<td>National horizontal and vertical</td>
<td>Global horizontal, vertical, and diversified</td>
</tr>
<tr>
<td></td>
<td>Concentration</td>
<td>Low</td>
<td>Increasing</td>
<td>Increasing</td>
</tr>
<tr>
<td></td>
<td>Internationalization of circuits of capital</td>
<td>Commodity capital</td>
<td>Money capital</td>
<td>Productive capital</td>
</tr>
</tbody>
</table>

Adapted and extended from J. Graham, Gibson, Horvath, and Shakow (1988, Table 1).
Figure 4.1: A framework for connecting capitalist variants, product cycles, and industrial districts across four periods of software.

Each software period has its own regional configuration, coinciding with the form of capitalist accumulation.
Calculator) was ENIAC’s electromagnetic counterpart and developed by International Business Machines (IBM) at Harvard University at the same time. Both were custom-built computers, enormous in physical size, and used to compute mathematical problems for the US military. Intelligence and military officials knew the computers had the power to solve complex intelligence issues like decoding and calculating ballistic trajectories, yet vastly underestimated that the complexity of telling a computer how to solve these computational problems. The “ENIAC girls” were a crew of 100 women, at its largest, that were responsible for running the ENIAC and whose official job title was “computer.” Management so misunderstood the complexity of programming the digital machine that the women programmers had to create their own social structures, labor processes, and divisions of tasks together, socially, on the shop floor. Computing’s “innovation” was in the use and application of the machine, but was misunderstood by management as simply the machine itself. When Douglas Hartree, the lead physicist behind ENIAC, published in Nature to debut ENIAC at the end of World War II in 1945, he focused on its hardware characteristics:

Its flexibility and speed of operation will make it possible to carry out many numerical calculations, in many fields of investigation, which without its assistance would have been regarded as much too long and laborious to undertake. (Hartree 1946, 546)

The use of these specialized machine-computers over the next decade made it clear to those that were making computational demands from the systems that the nature of operating the machine was anything but clerical. Giving instructions to the computer required a great deal of logic and coordination amongst the programmers. In fact, working with the SAGE system is what forced MIT, IBM and the US Armed Forces to distinguish between specifying the system (as a systems analyst) and programming the system (as
a computer programmer) (Kraft 1979, 144). New demands and applications of the ENIAC and Mark I and the new computers like the iterations of the Universal Automatic Computer (UNIVAC) and the Semi-Automatic Ground Environment (the SAGE missile defense) system, required increased efficiency on the part of the entire productive system.

### 4.3.2 Software contracting

After successfully completing a contract to research and develop the Mark I at Harvard University, IBM was contracted in 1958 by MIT (and by proxy the US Air Force Cambridge Research Laboratory) to develop the SAGE missile defense computers and computer network. IBM’s portion of the SAGE missile defense system amounted to almost $500 Million USD, or 56 computers for roughly $18 million per computer (Earnest, Wong, and Edwards 1998). These contracts helped catalyze software production in the 1950s and 1960s by making IBM the de facto, near monopoly, computer manufacturer. This is not a single event per se, but part of a series of events that had snowball-like impacts in IBM’s success. The experience developing these computers translated directly into developing future computers, and, as the Backus quote above demonstrates, became a space for innovation.

In the 1950s and 1960s, computer manufacturers also bundled software services with computer sales, requiring that computer manufacturers engage in software production activities. Once a buyer purchased a computer from IBM, programmers were also required to operate the computer and change programs per the changing requirements of the organization. Firms – both computer manufacturer and computer purchaser – saw programming as the single largest source of labor in the process and the easiest place to introduce management. But, firms also saw that standardizing the operation of computers meant that programmers didn’t have to learn new instruction sets as computers evolved.
4.3.3 Innovation and the rationalization of labor processes

The two needs found a common resolution in two influential events that happened in the 1950s and matured in the 1960s, both involving IBM. The first event happened in 1957 when IBM released one of the earliest computer programming languages compatible across computers (mainly, the various IBM computers). Formula Translator, or FORTRAN (now Fortran) compiled to the specific assembly language of the computer architecture. Programmers could write instructions in Fortran, then run a compiler to optimize the program and create an assembly language program for direct use on the intended computer architecture. The second event was the development of a computer that could run stored programs. IBM developed the IBM 1401 computer which was the first computer to use stored-computer programs and be fully transistorized. Later in the 1960s, IBM introduced the System/360 which could run Fortran and a new language called COBOL. For now, the IBM 1401 represented a major advantage in flexibility, partially because it could run Fortran and stored Fortran programs. On Fortran, IBM touted its business applicability.

The result should be a considerable reduction in the training required to program, as well as in the time consumed in writing programs and eliminating their errors. (IBM 1956, 4)

The lead developer behind Fortran, John Backus, said in Think, the IBM employee magazine, that the impetus to make Fortran came from his own frustrations.

Much of my work has come from being lazy. I didn’t like writing programs, and so, when I was working on the IBM 701, writing programs for computing missile trajectories, I started work on a programming system to make it
easier to write programs for the 701. (John Backus interviewed in Stegmann July/August 1979, 21)

IBM even proclaimed that it could reduce the total number of instructions needed for programmer to write by a factor of 20. Firms adopted the language with these efficiencies in mind but it took until the mid 1960s before Fortran achieved widespread adoption.

As the use of large computers spread from academia and the military to industry through computers developed by companies like IBM, large firms with rigid bureaucracy and hierarchy attempted to rationalize the labor process of programming. Programmers in service of academic and military labs manufactured software by programming directly on the computer through buttons and switches. Returning to a key characteristic of software, and referencing Backus earlier, software’s production relies on existing software. One way to create efficiency is to create a higher-level language that writes lower-level code with fewer coding statements. On the first generation of digital computers used in industry, programmers wrote software in assembly language that was specific to the processing architecture of that computer. As the industry matured, primarily through IBM, programmers could write Fortran and COBOL code that compiled into assembler code. And importantly, the new possibilities available to firms because of software and computing development translated into growth of software production activity and computer hardware manufacturing.

The transformation of computing into industry alters software labor into a ‘purely objective productive’ purpose.

In manufacture the organisation of the social labour process is purely subjective, and is a combination of detail workers; in machinofacture, large scale industry has a purely objective productive organism, in which the worker is nothing more than an appendage. (Marx 1976 [1897], 480)
Software contracting emerged as the variant of capitalist production, and the objective management practices adopted from software company’s parents firms introduced hierarchy, bureaucracy, and rationalization within the workplace. That is, software production activity within the firm renders labor part of the ‘productive organism.’

### 4.3.4 Growth of programming as a profession

The transition from military to industry, and rationalization of the programming workplace, produced a new relation between software production activity and the programmer. Most programmers during the 1950s and 1960s time were not trained to be programmers, but learned to program because their job necessitated it. Programmers elected or preselected to become programmers. ENIAC and Mark I programmers were primarily women because of the perceived clerical nature of button-pushing. Through the 1960s, programming, along with engineering, became more selective and the public perceived programming to be math and science heavy as computing grew. This led to a gender preselection among potential workers, but Greenbaum (1979, 87) points out that despite this preselection, general prejudice was the largest barrier for women entrants, and likely minority entrants, into the emerging software labor market. Many of the women who were in computer programming before software production activities diffused from the lab to industry, prior to the 1950s and 1960s, stayed in these roles. During the 1950s and the early part of the 1960s, women programmers were seldom hired (Reskin and Roos 1990).

The rationalization of the workplace in the 1960s meant that programming within the firm (and software production activity in general) was constantly undergoing reorganization by management. One outcomes was the splitting existing the programmer into low skilled and high skilled specializations. In the late 1960s and early 1970s, the rapid expansion of software programming and the narrow(ing) definitions of the low skilled
occupations meant that firms again hired women programmers (Reskin and Roos 1990; Greenbaum 1979).

The number of programmers necessary to develop a single instruction set meant that the labor of programming required social interaction, cooperation, and collaboration especially on the ENIAC and early computers where there was a distinct lack of management. Computers themselves were no longer confined to research and militaristic purposes, and instead produced and sold as commodities (fixed-capital) for large businesses. Programming for these companies required different software programs for each installation – sometimes altogether different and sometimes marginally different. New software contracting companies formed to provide software at lower rates than computer hardware makers by leveraging their in-house expertise to reuse code in new situations. There were still few programmers and software contracting companies had the capacity to concentrate laborers and code collections for this purpose.

4.3.5 Emergence of corporate production and the dialectic of software production

As Campbell-Kelly 2004 points out, the production of software as a commodity took the form contract work in the 1950s and 1960s. As a business to business interaction, capital-capital, the relationship worked. Purchasers of computers also received the software necessary to operate them. Computers were still large and expensive, but IBM began introducing a series of computers that were not customized, and could run large-scale software programs – effectively creating large-scale software commodities market. Software firms competed through scope narrowing, budgeting, and the rationalization of labor. The rationalization of programming organizational structures meant that management organized workers based on task or project which allowed management to leverage organizational charts to identify accountability, measure development speed, and intro-
duce predictable development actions (Greenbaum 1979, 110-11). Greenbaum (1979) interviewed a system analyst that expressed their reactions to reorganization.

They [top management] keep changing the organization chart. The name of the game is change the structure. We’ve come to accept yearly restructuring—they just keep shuffling managers around. At least once a year there is some major productivity crisis and their response is to reorganize us into different groups and make the groups report to different managerial functions. We’ve come to takes these changes as commonplace. Sometimes we don’t even talk about it; we just go on doing what we always do. (interview subject quoted in Greenbaum 1979, 114)

Firms were aware of the expensive cost of programming labor. Firms that developed computers and software contracting firms sought ways to reduce the expense of labor and software production activities. The logic of efficiency permeated into the development of software itself. The development of Fortran and the increasing management were not unique phenomenon to the contexts of the above quotes. In the 1960s, software services firm Informatics developed the MARK IV programming language (unrelated to the Mark I computer). The MARK IV sales pamphlet speaks to the same time-saving concerns and promises of efficiency that IBM’s Fortran promised.

With MARK IV we can do the 100% workload of the programming team with 40% of the team. The available 60% is now devoted to development. (Customer testimonial in Informatics Incorporated 1971, 7)

MARK IV is an example of a software language that abstracts instructions to fewer commands, as outlined in Table 4.3. MARK IV, Fortran, and COBOL perform this task and higher-level than the 2nd generation code that they produced through “compiling.”
Informatics developed MARK IV using the same 2nd generation languages it sought to abstract. The market for MARK IV, and other types of languages, was strong allowing Informatics to license MARK IV to firms who already had programmers. Licensees then generated new software programs. Here is one of software’s key characteristics: programmers require existing software to develop new software. Since the first software programs involved direct manipulation of the computer’s bits through button pushing, the first forms of automation. Yet, as the customer testimonial suggests, that does not necessarily translate to deskilling of the software worker.

Table 4.3: Software programming language generations.

<table>
<thead>
<tr>
<th>Generation (GL)</th>
<th>Hardware</th>
<th>Example languages</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Machine language</td>
<td>Bit programing / button pushing</td>
<td>Machine</td>
</tr>
<tr>
<td>2nd</td>
<td>Machine specific assembler</td>
<td>Machine specific assembler</td>
<td>Low</td>
</tr>
<tr>
<td>Early 3rd</td>
<td>Machine independent compiled</td>
<td>Fortran, COBOL, C, C++, C#</td>
<td></td>
</tr>
<tr>
<td>Advanced 3rd</td>
<td>Machine independent, not necessarily compiled</td>
<td>Python, Ruby, Perl</td>
<td></td>
</tr>
<tr>
<td>4th</td>
<td>Rapid language assisted prototyping</td>
<td>Libraries exist for 3GL</td>
<td></td>
</tr>
<tr>
<td>5th</td>
<td>Constraint-based AI programming, machine-learning</td>
<td>LISP, TensorFlow (library)</td>
<td>High</td>
</tr>
</tbody>
</table>

Adapted from IBM (2010) and Miller, Vandome, and McBrewster (2010).

Other testimonials in the MARK IV sales pamphlet are also illuminating.

We assigned a programmer, one of our sharpest, to a job which looked like a job best done in COBOL. The programmer was told to do the job in COBOL and was given six months for completion. On his own, the
programmer secretly did the job using MARK IV and completed the job in 3 weeks. (Customer testimonial in Informatics Incorporated 1971, 5)

Fortran’s manual contains similar claims to those in the MARK IV sales pamphlet.

Object programs produced by Fortran will be nearly as efficient as those written by good programmers. (IBM 1956)

Fortran doesn’t necessarily deskill the worker who created the Fortran code, but instead deskills other workers, some of them software workers who are competing in the labor market.

high-level languages and appropriate translators obviously requires advanced skills and knowledge on the part of the language designer and programmers. On the other hand, once that slow and expensive task is completed, the products can be used by anyone who has mastered the ‘alphabet’ and ‘syntax’ of the simpler high-level languages. In short, highlevel languages make it unnecessary for programmers to be quasi [hardware] engineers. (Kraft 1979, 146)

The adoption of higher-level languages allow the firm to apply a software worker’s labor in others areas of programming in service of the firms expansion and future profit-seeking. Software programming deflects deskilling from the software producer and instead directly contributes to the general deskilling of the workforce. The software worker, after all, is the one that produced the new software and may be the (only) one that understands its construction, its necessary maintenance, and how to modify it for future applications. In this sense, the software acts as fixed-capital, machinery to be used as a means of production, for software users. Software workers, neither deliberately nor directly, always compete to deskill each other. The reapplication of deskilled software
workers with new technologies in the pursuit of firm efficiency contributes to the social acceleration of competition among firms.

In the next section I will detail the developments that alter this form of production and the commodity form of software. In the 1950s and 1960s, computing’s diffusion into industry introduces software as a corporate commodity, from, Table 4.1, but also introduces a transition into what J. Graham et al. (1988) calls the “competitive” variant of capitalist production in Table 4.2. Software contracting relied on redundant labor and machinofacture forms of production. Yet the spatially diffuse software contracting lacked the flexibility for wider adoption by firms.

4.4 1970s and 1980s: Corporate Software, Memory and Databases

Software contractors developed computer architecture specific software in the 1950s and 1960s. Software contracting firms sought economies of scale in the development of software. Firms sought to reduce labor costs through automation and through increased scientific management of software production activities. Software contracting firms, and software producing firms in general, faced two significant challenge. How could they create a software that would run for different clients on each of their differing computing architectures? Would there be a significant enough market for standardized software? This section outlines three series of events that catalyzed the corporate software era of 1970s and 1980s. Then it discusses the rise of a new profit cycle and the conditions that led to the period where software emerged as a mass-market commodity.

4.4.1 IBM innovations and constraints

The first series of events are mainly technological and represent the technological developments of IBM. Magnetic tapes and disks began to replace punch cards, increasing the storage capacity for both computer memory and data storage. Magnetic tapes allowed for
large storage capacities. IBM’s introduction of the 8-inch floppy drive (officially the IBM 23FD) started the floppy drive revolution leading to the 3.5-inch floppy drive released in 1982. Fortran and COBOL were among the most widely adopted programming languages and firms created compilers for different system architectures. These events became relevant together when, in 1970, IBM introduced relational databases, the ability to lookup and store data for programmers to synchronize between different users and/or terminals. The System/360 cornered the market because it had large internal memory, and the ability to connect to both magnetic tape systems for relational databases and floppy drives for the distribution of software. While introduced in the mid to late 1960s, the System/360 saw its success in 1970s. The system was so successful that programmers primarily focused on standardizing on the architecture of the System/360. Today’s computers are direct descendants of this architecture.

In 1969, IBM, under anti-trust pressure from the US Department of Justice claiming that IBM was attempting to monopolize the general purpose electronic digital computer system market, decided to famously “unbundle” its operating system from its computers. Firms could buy computers and they could buy software. Firms were already purchasing software under software contracting agreements during this time, but this event opened the market to any competitor. Further, it stopped manufacturing its own silicon memory and decided to contract out these components. Intel was the first company to produce a commercially available microprocessor in 1971, and IBM used an Intel processor in its typewriters in 1972, before standardizing on the Intel 8080 and 8086 processors, which are the basis for existing processor architecture (x86).

4.4.2 Economic recessions, market changes, and the software commodity

The second series of events start with the short recession of 1969-1970 and the coinciding decision by the Federal Reserve to raise interest rates meant that firms had less cash to
Figure 4.2: IBM’s market share and projected growth. Fortune Magazine, 1968.

From Burck (1968).
buy expensive computers, software, and computer programmers. Despite the market not playing out as expected between 1968 and 1972 – see Figure 4.2 from Burck 1968 – IBM still did well: firms sought lower cost options, like that of the comparatively inexpensive and versatile IBM System/360. In 1968 software firms were among the best performing stocks in the US stock markets and software firms fared well at the beginning of the recession. Computer companies, after the recession, felt continued pressure as potential computer buyers delayed their orders or canceled them altogether. Firms sought cheaper computers and firms also sought to buy less expensive software. IBM’s decision to unbundles software from its computers ushered a new era of software development, and speculation in software firms.

To be clear, the first software bubble was not the dot-com bubble of the early 2000s, but instead the later half of a recession from 1969-70 that coincided with a broader economic recession. Ross Perot famously lost $450 million in a single day due to the recession and the change in his software company’s stock. The unclear and unfamiliar nature of the computer and software market meant volatility for investors (Brooks and Lewis 1999; Campbell-Kelly 2004). Software contracting companies had emerged, but the new flexibility to create and market software for any number of versatile machines allowed software producers and consumers great capacity for innovation. In the late 1960s, prior to the recession, Applied Data Research (ADR) built a prototype flowcharting program, Autoflow, for Radio Corporation of America (RCA). RCA lost interest in the product, and ADR focused on licensing Autoflow to new customers. They focused on firms with the IBM 1401/1410, allowing them to standardize their software for the platform. ADR filed for a patent on their software, the first patent for a software program. This marks the first time that producers of software sold software as a commodity, rather than packaged as a computer, or as custom software services (Campbell-Kelly and Garcia-Swartz 2015).

ADR’s Autoflow is an exception in the 1960s. Autoflow made ADR extremely
successful in the 1970s when firms sought cheaper capital goods. Autoflow operated primarily on the IBM 1401, and the IBM 1401 did not have an operating systems by todays standards. A single program could run on the computer at a time, reducing the overall flexibility that firms required. IBM’s decision in 1969 to unbundle its software from its computers and IBM’s standardization on the Intel x86 architecture allowed for Microsoft to license a DOS (disk operating system) to IBM. IBM had already developed an operating system for its popular System/360 named OS/360, yet OS/360 wasn’t necessarily compatible with the x86 architecture. IBM offered several different operating systems to their clients. Microsoft’s operating system was by far the cheapest of the offerings and compatible with Digitals Research’s Control Program/Monitor (CP/M) which was the current market standard. Microsoft and Intel retained the rights to sell their operating system to other computer makers. These events unfolded throughout the 1970s and by the 1980s, IBM was selling PC-DOS, a repackaged Microsoft DOS (MS-DOS), on their computers running with an Intel x86 architecture.

4.4.3 The spreadsheet and the operating system

The third series of events relate to the operating system and the spreadsheet. The corporate software era of the 1970s and 1980s saw first movement of computers use move from specialized machinery to the desktop. This move happened within large firms, and these firms were still the main clients of the expensive desktop computer. Desktop computers were less expensive than their mainframe counterparts. But during this move, the personal computer, made by any number of computer manufactures, standardized running PC-DOS on the Intel architecture and provided a single platform for firms to create software. VisiCalc, one of the most popular early software programs, was the IBM produced spreadsheet program that was the business standard until Lotus developed 1-2-3 which, at a lower price point, out-competed and outperformed VisiCalc. Through a
series of chance opportunities and path dependence, the confluence of Intel’s architecture, Microsoft’s operating system, and Lotus’s software program resulted in a set corporate computing standards. This configuration benefited from network externalities – the characteristic that the software or product gains value with larger adoption.

This history has primarily focused on the IBM PC, the standard within firms whose legacy is dominant today. However, Apple released the Macintosh in 1984, Microsoft released Windows in 1986 (which eventually supplanted MS-DOS), and an obscure programmer released linux in 1991. Each of these platforms challenges to the dominant desktop computing platform.

During this time, new companies in California, like Intel, benefited firm and government based research. The US military and federal government awarded the majority of contracts through the 1960s to large east coast firms (Saxenian 1996). In the late 1960s, new competitors like Hewlett-Packard, research institutions like Xeroc PARC and Stanford University, contributed heavily to the geographic shift toward the west and Silicon Valley in particular (Saxenian 1983). Visits from Steve Jobs and Bill Gates to Xerox PARC cemented the idea of a graphical operating system, both later developed into the modern standards of desktop operating systems.

This period, and the growth of MS-DOS, marks the initial decline of IBM from software in general. Autoflow and VisiCalc were initial forms of software commodities that were primarily used in corporate software. Software’s commodity form hadn’t entirely caught on. The video game industry, in the late 1970s until its bust in 1983, provided the vision for independent software makers that could sell software as a mass-market commodity – software on a diskette could be like a video game cartridge bought in the store. The video game industry had lasting impacts on software production. The extravagant campuses of video game makers and their employees influenced the corporate campuses of software firms and the expectations of software workers (Wolf 2008).
4.4.4 Labor and new programming languages

The US Census Bureau first created software specific occupation codes in 1967 for use in the 1970 Decennial Census (Markusen 1986). Most software production during this time happened under three different software occupation codes: the computer programmer, the computer software analyst, and the operations and research analyst. These occupations went unchanged in the 1980 and 1990 census, yet the number of workers in software production activity went from almost 300,000 in 1970 to nearly 1.3 million in 1990. Software production activity accounted for 0.35% of all labor in 1970. By 1990, it had tripled its share to 1.03% (see Table 4.5). The firm organization and business practices surrounding software production in the 1970s, 1980s, and 1990s mirrored the initial formations that followed the transition of software production from military and universities to industry in the 1950s and 1960s. While firm organization and practices persisted, divisions on the “shop floor” of software work began to emerge and software occupations began to splinter (Greenbaum 1979; Reskin and Roos 1990). Kraft (1979) documents this splintering in the 1970s.

The creation of two software occupations, where before there had been only one, was to be the first of a long series of subdivisions. The analyst/programmer distinction was, and to a large extent remains, crude, tentative, and spurious. Yet it proved that programming could be divided into two main categories of more thoughtful and less thoughtful work. Even relatively ‘creative’ software specialists, i.e., specialists whose work involved little mechanical detail, could now have their work routinized. The fragments thus created could be parcelled out to less skilled workers, e.g., technicians, or still another new suboccupation, ‘coders.’ Managers at last could begin to replicate in programming the finer divisions long established in electrical
engineering and other more ‘traditional’ engineering occupations. (Kraft 1979, 145)

Programming was proliferating into the workplace as software workers nearly quadrupled their presence. At the same, the corporate era of software production also saw that more office workers begin to interact, some even making their own software programs to run on mainframes or desktop computers.

The growth of software production activity, both explicitly as a programmer and as a skill, meant that there was also a demand for workers with any level programming experience. Computer science became a college degree that equipped workers with the skills necessary to fill software occupations and training courses became available for existing workers to gain software skills. Even still, software was still little known as a career path. Software’s importance and future growth were still unclear within mainstream society and considered a boutique industry and skill.

My intention at the time, I still new nothing about computer science, I was a computer science major and didn’t know anything about the big picture, I didn’t take it seriously. This is a way to pay my way through as an undergrad-uate, then after I graduate I’ll go back to what I really care about, I’ll go back to history, or to a political science major, I’ll join the State Department, I’ll do something completely different. There were people that I was in school with that talked about things like getting jobs in San Jose at the time. I graduated in 1983, and I had no idea what they were talking about. I thought going to California was a good idea, but Silicon Valley didn’t mean anything to me.²

²Austin developer on programming in the 1980s. Interview, 10/31/2015, D9, transcript.

Even then, it’s perception as being math and science heavy turned would-be recruits away.
An Airforce recruiter said I could get an ROTC scholarship if I switched to computer science. I said no. But then I got desperate enough... I had the aptitude for it, but I didn’t care for it... I was programming in Fortran and assembler.³

The computer system analyst was responsible for not only programming, but translating business needs into problems that software workers could solve. Computer programmers were the army whose responsibility was to produce the programs with the aide of systems analysts. Operations and research analysts either operated or research specific computer platforms. The rationalization of software labor took on the construction-industry characteristics of the previous software period, perceiving software to be a concretely specifiable subject and object of labor like that of constructing a building. Software programs and products did not behave as passively as buildings once constructed and as divisions within on the shop floor emerged, managers reconfigured the “assembly line” and post-production support to mimic electrical engineering.

Present-day programmer training, although institutionally separate from electrical engineering departments, almost exactly parallels the structure of the previously developed model. The lowest level specialists-coders and applications programmers-are trained in junior or community college programs, designed to prepare them for technician jobs in local industries. More skilled programmers-people who work on complete programs rather than on fragments of a single program-are trained in conventional engineering colleges. Occasionally, they are trained in liberal arts colleges, in which case they may receive degrees in applied mathematics as well as in computer science. (Kraft 1979, 150)

³Austin product manager on programming in the 1980s. Interview, 10/30/2015, D7, transcript.
New specifications of software languages accelerated the division of software occupations as those with experience in newer software languages could achieve higher labor efficiency than their peers. Management relied on heavy up-front design to divide programming tasks, bifurcating and dividing high-skill from low-skill work.

The corporate software era saw the languages like C and C++ make the same claims that Fortran, COBOL, and MARK IV made in the previous software period.

At [California firm], it was a pretty boring programming job, but I found it to be easy. Very easy. Low stress. COBOL and a language called MARK IV. It was real top heavy with management.⁴

The increasing demand of software workers tempered the persistence of rigid working conditions, new expectations from new languages and management styles, and the continuation of prejudice in hiring and the workplace. New opportunities for women and minorities came primarily from the fact that software workers were in demand and the existing divisions of labor within each occupation and between occupation classes were already established.

4.4.5 Emergence of mass-marketing software

In summary, firms relied on scientific management and began extend the use of software languages in the pursuit of automation. The 1970s and 1980s also mark the increasing concentration of software workers, and the national competitive nature of large computing firms. These track with the monopoly form of software production. In the 1970s and 1980s, the standardization on MS-DOS, and the standardization on Intel’s x86 architecture, set the stage for software to be a mainstream, mass-market commodity. The growth of software production activities across industries and the development of computer science

⁴Austin developer on programming in the 1980s. Interview, 10/31/2015, D9, transcript.
degrees provided the labor. The profit cycles afforded by corporate software would dwarf compared the new market opened up because of the standardized computing platform – whose size made it possible to put on desks. The result was new software programs that provided efficiency in the workplace and home were in demand. The video game industry influenced software’s mass-market form. Further, new sets of social relations (Saxenian 1996) and demonstrated success of video game programmers (Wolf 2008) attracted new laborers to software production.

4.5 1990s and 2000s: Mass-market Software, Desktop Computers, AOL

With the growth of software and computing comes a growth of software and computing events – and the events of the 1990s and early 2000s are too voluminous to document here. Instead this section, with a focus on labor processes, organizational strategies, and technological developments will show how this period is markedly different from previous periods because of the rapid growth and diffusion of software into society.

Table 4.4: Growth of software labor, 1970–2015.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Software workers</td>
<td>291,000</td>
<td>539,960</td>
<td>1,288,984</td>
<td>3,294,397</td>
<td>3,618,295</td>
<td>4,244,971</td>
</tr>
<tr>
<td>Percent increase</td>
<td>86%</td>
<td>139%</td>
<td>156%</td>
<td>10%</td>
<td>17%</td>
<td></td>
</tr>
</tbody>
</table>

4.5.1 A computer in every home, operating systems

In a Microsoft news conference in 2006, Bill Gates recalled his original business goals.

When Paul Allen and I started Microsoft over 30 years ago, we had big dreams about software. We had dreams about the impact it could have. We talked about a computer on every desk and in every home. (Beaumont 2008)

The personal computer epitomizes the 1990s and early 2000s. Demand for computers with firms and homes grew, and with it the price and size of computers dropped. The x86 architecture dominated all but the primarily Motorola and Apple-backed RISC architecture of the minuscule non-IBM PC market. With a narrowing set of operating systems, third-party software for productivity, education, or entertainment became standard – hardware makers stopped making software specific for their platforms.

Table 4.5: Software workers share of in industry employment, 1970–2015.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Software workers</td>
<td>291,000</td>
<td>539,960</td>
<td>1,288,984</td>
<td>3,294,397</td>
<td>3,618,295</td>
<td>4,244,971</td>
</tr>
<tr>
<td>Average share of industries</td>
<td>0.51 %</td>
<td>0.56 %</td>
<td>1.06 %</td>
<td>2.03 %</td>
<td>2.09 %</td>
<td>2.31 %</td>
</tr>
<tr>
<td>Share of all workers</td>
<td>0.35 %</td>
<td>0.51 %</td>
<td>1.03 %</td>
<td>2.37 %</td>
<td>2.36 %</td>
<td>2.66 %</td>
</tr>
<tr>
<td>All workers</td>
<td>82 million</td>
<td>106 million</td>
<td>125 million</td>
<td>139 million</td>
<td>153 million</td>
<td>160 million</td>
</tr>
</tbody>
</table>


Software became a retail sensation. Retailers like Egghead Software sold only software products, and existing retailers added their aisles devoted to only to software products.
Compact discs (CDs) and large disk drives meant that users could cheaply store large volumes of installed software on their computers – providing new distribution opportunities for software products. In the US, software became a mass-market commodity.

While Microsoft’s series of Windows operating systems, all built on or run on Microsoft’s DOS operating system, dominated the 1990s and 2000s, Apple emerged as Microsoft’s largest operating system competitor. Microsoft’s operating systems were primarily for IBM lineage of Intel’s x86 architecture. New chip manufactures competed against Intel’s dominance of the microprocessor and companies like Cyrix and Advanced Micro Devices (AMD) competed with their own x86-compatible architectures. Linux, made entirely open source, could run on any number of architectures and was a free alternative to expensive Apple computers or expensive licenses to Microsoft operating systems. Apple, realizing the cost burden of maintaining its own architecture and own operating system while competing with behemoth Microsoft, sought to take advantage of the network effects of software and hardware. In the early 2000s, Apple adopted a linux variant as the base of their tenth version of their Mac Operating System (commonly Mac OS X, now macOS) and then moved entirely to Intel processors.

This rushed account of the changes in architecture and the rapid diffusion of the personal computer is only part of the story of the 1990s and 2000s. With a computer in every home, the ability to connect them spurred the development of the modem and the internet. This introduced new forms of person to person, business to customer, and business to business relationships. Firms could sell to customers online and firms could perform customer relationship management tasks on the new medium. New exploratory forms of social interaction, from chat rooms to virtual reality, brought with it new forms of software commodities.

A computer in every home made possible by the decreasing costs of computer hardware, the increased distribution data capacity provided by CDs and the fledgling internet,
the growth of the concept of software as a mass-market commodity, and the growing ubiquity of computing spurred software’s economic growth. The economic growth of software in the home, and the application of software within the firm for logistics, meant the development and sale of software demanded labor to create, modify, or maintain software and software installations. Between 1990 and 2000, the growth of software workers wasn’t limited to industries like data services or computer manufacturers. The year 2000 marks a moment when software diffused into industries, including many in the manufacturing, transportation, wholesale trade, finance, business, and professional industry groups as shown in Figure 4.3.

4.5.2 Browsers, search, and the new Silicon Valley

The connectivity of computers and their users brought new attention to internet software like the browser, run on a user’s computer, and portals, which ran on servers from internet service provider firms. The internet browser software and the starting location of the browser became the center of two important competitions. Netscape Navigator was the standard internet browser, securing over 90% of the browser market share, until Microsoft began bundling Internet Explorer with its dominant Windows operating system. By the fourth version of Microsoft’s Internet Explorer, little incentive existing for users to spend the time or money to download and install a competing browsers. This bundling move by Microsoft further solidified Microsoft’s dominance in controlling the the software that ran on the growing number of desktop computers. As Netscape declined, America Online (AOL) became Microsoft’s largest competitor because it served as browser, portal, and internet service provider. Installing and using AOL did not require lengthy downloads because AOL sent free physical copies of its browser to customers in the mail with free internet hours. By 1999, AOL bought the failing Netscape.
The history of the browser sets in motion two catalyzing socio-technical events. First, Microsoft’s bundling of its own browser with its own operating system began nearly decade long legal battle with the United States over anti-competitive practices. This emboldened the Free and Open Source Software (FOSS) movement and led to a growing backlash against Microsoft. Second, Netscape released the source code to its browser in 1999, immediately before AOL’s acquisition. FOSS developers took Netscape’s source code and developed Mozilla Firefox. Firefox’s success in competing with Microsoft provided hope for open-source developers and led to the growth of open-source software firms. Importantly, Mozilla defaulted its web portal and search engine to a Google Search. Google, founded in 1998, swept across the internet with the rapid growth of Firefox, becoming the default search engine for the majority of the internet, effectively eliminated other players like AOL, Yahoo, and AltaVista. Apple had its own resurgence with that coincided with its the release of its newest operating system, yet Microsoft had a version of Internet Explorer and its Office productivity software packages for Apple platforms.

The success of Bay Area hardware, software, and web firms like Google, ecommerce firm eBay, Apple, database software firm Oracle, and network hardware firm Cisco, continued to solidify new labor markets outside the traditional east coast metropolitan areas. The industrial diffusion of software workers met with the spatial diffusion of software workers. In unsuspecting places like Seattle, Microsoft, media firm Real Player, and fledgling ecommerce firm Amazon, each grew to become dominant players in their respective markets. Software presented such new opportunity that even IBM exited the personal computer industry to focus on large mainframes and software services.

4.5.3 Growth, bust, innovation, startups

In the late 1990s, software’s explosion into the mass-market provided opportunities for programmers and entrepreneurs to create software products to appeal to large audiences.
Figure 4.3: Diffusion of software workers across selected industries, 1970–2015.

### INDUSTRY GROUP
- **MINING**
- **MANUFACTURING**
- **TRANSPORTATION, COMMUNICATIONS, AND OTHER PUBLIC UTILITIES**
- **WHOLESALE TRADE**
- **RETAIL TRADE**
- **FINANCE, INSURANCE, AND REAL ESTATE**
- **BUSINESS AND REPAIR SERVICES**
- **PROFESSIONAL AND RELATED SERVICES**

#### SOFTWARE WORKERS
- 42 Oil and gas extraction
- 390 Toys, amusement, and sporting goods
- 380 Photographic equipment and supplies
- 372 Medical, dental, and optical instruments and supplies
- 371 Scientific and controlling instruments
- 370 Cycles and miscellaneous transportation equipment
- 362 Guided missiles, space vehicles, and parts
- 352 Aircraft and parts
- 342 Electrical machinery, equipment, and supplies, n.e.c.
- 341 Radio, TV, and communication equipment
- 340 Household appliances
- 332 Machinery, n.s.
- 322 Computers and related equipment
- 321 Office and accounting machines
- 312 Construction and material handling machines
- 311 Farm machinery and equipment
- 310 Engines and turbines
- 292 Ordnance
- 221 Footwear, except rubber and plastic
- 200 Petroleum refining
- 192 Industrial and miscellaneous chemicals
- 191 Agricultural chemicals
- 190 Paints, varnishes, and related products
- 182 Soaps and cosmetics
- 181 Drugs
- 172 Printing, publishing, and allied industries, except newspapers
- 130 Tobacco manufactures
- 472 Utilities, n.s., and other combinations
- 452 Electric and gas, and other combinations
- 451 Gas and steam supply systems
- 450 Electric light and power
- 442 Telegraph and miscellaneous communications services
- 441 Telephone communications
- 440 Radio and television broadcasting and cable
- 552 Petroleum products
- 541 Drugs, chemicals, and allied products
- 530 Machinery, equipment, and supplies
- 512 Electrical goods
- 511 Metals and minerals, except petroleum
- 510 Professional and commercial equipment and supplies
- 663 Catalog and mail order houses
- 633 Radio, TV, and computer stores
- 712 Real estate, including real estate–insurance offices.
- 711 Insurance
- 710 Security, commodity brokerage, and investment companies
- 702 Credit agencies, n.e.c.
- 701 Savings institutions, including credit unions
- 700 Banking
- 752 Electrical repair shops
- 741 Business services, n.e.c.
- 732 Computer and data processing services
- 721 Advertising
- 893 Miscellaneous professional and related services.
- 892 Management and public relations services
- 891 Research, development, and testing services
- 890 Accounting, auditing, and bookkeeping services
- 882 Engineering, architectural, and surveying services
- 881 Membership organizations, n.e.c.
- 850 Colleges and universities

#### Percent software workers
- > 7.18%
- > 2.53%
- < 2.53%
- None

103
Larger firms sought to protect their market dominance and acquired new firms that had successful products or innovative technologies. Venture capitalists sought to grow software firms quickly. Speculation fueled investment in software and web related firms, signaling a new direction for stalwarts of the hardware industry. In the 1990s, the focus shifted from hardware and computer firms to software firms and software services.

The growth of the mass-market software period was not without its own bust. The NASDAQ stock exchange, home to the high-technology and software firms of the dotcom boom and long a gauge of the HT/IT industries, reached its peak of over 5,000 points during March of 2000. A decade earlier, in March of 1990, the NASDAQ Composite Index had a value of around 450 points. The character of the NASDAQ index’s growth is not of slow and steady climb. In March of 1999, the index had risen to nearly 2,500 points. From this new record high, it doubling in one year to reach it’s new high in March of 2000. One interviewee captured the excitement of the of the time:

Everyone thought the new economy, the dotcom thing, was different. Everyone said it. Even your taxi driver was recommending which internet stock to buy. Everyone was caught up in that mania.\(^5\)

The growth of the number of software workers, the demand for software in the home, the demand for software within the firm, combined with the industrial and spatial diffusion of software skills lay prime conditions for innovations. Workers seeking to take

\(^5\) Austin developer and entrepreneur. Interview, 11/02/2015, D11, transcript.
advantage of this growth learned to program on their own, through a new programming book industry, or through new forms of certification. The dot-com bubble burst, sending technology stocks into a recession. Coupled with the recession after September 11th, 2001, software and technology firms consolidated and reorganized.

By March of 2001, the NASDAQ index had fallen to less than 2,000 points, signaling a massive decline in investor confidence in software as a market, a commodity, and a skill. The NASDAQ reached a low in late 2002 of around 1,200 points. The structural changes as a result of diffusion of software within firms and labor markets had already taken hold. Using the average share of software workers in each industry for 1970, Figure 4.3 tracks industries that had a standard deviation above the 1970 average (2.53%) during any of the time periods of 1970, 1980, 1990, 2000, 2005–2009, and 2011–2015. Starting in 2000, software workers steadily occupied larger portions of industries than initial representation in the 1970s.

Building from the oft used measures of industrial change, the coefficient of specialization and the coefficient of localization Isard (1956), Table 4.6 shows the calculation of a coefficient of industrial specialization measuring the share of software workers by industry. The coefficient of industrial specialization (CoIS) uses the equation

\[
\text{Coefficient of Industrial Specialization} = \frac{1}{2} \sum \left| \frac{W_{occ,ind}}{W_{occ}} - \frac{W_{ind}}{W} \right| \quad (4.1)
\]

where \(W_{occ,ind}\) is the number of workers of a given occupation (in this case, software occupations) in a given industry, \(W_{occ}\) is the total number of workers with that occupation across all industries, \(W_{ind}\) is the total number of workers in any occupation within a given industry, and \(W\) is the total number of workers in any occupation across all industries. CoIS values closer to 1 signal perfect concentration (all software workers in a signal industry) while values closer to 0 signal equal shares of software workers in
Figure 4.3: Dotcom boom and bust, NASDAQ Composite INDEX (IXIC) 1997 to 2017.

IXIC NASDAQ COMPOSITE

Points
Reaches 5,048 on Mar 10, 2000
Surpass all-time high, nearly 6,100 in May 2017
Falls roughly 80%, bottoms out Oct 2002
Falls again, nearly 50%, during financial crisis

Jan 1997 to May 2017

Author’s charting from NASDAQ price and volume data.
each industry (varying by the size of the industry). The CoIS, calculated using software as the occupation of choice in 4.6, appears to trend to zero. This tracks with events in the software periods. In 2000, the CoIS reaches its lowest value, meaning software workers were most diffuse across industry groups. The dotcom bust followed by the beginning of the Great Recession in the 2005–2009 period drove software workers back toward concentration. The nature of this concentration post-2000 is different than the concentration pre-2000, software workers grew in industry groups like Finance, Real Estate, and Insurance and other business oriented industries (see Figure 4.3).

Table 4.6: Measuring concentration of software workers by industry, 1970–2015.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient of Industrial Specialization</td>
<td>0.404</td>
<td>0.382</td>
<td>0.362</td>
<td>0.325</td>
<td>0.359</td>
<td>0.361</td>
</tr>
</tbody>
</table>


This suggests that in the 1990s, the rapid growth of software was primarily in industries that have low numbers of software workers. This also had the result of software firms structuring the relationship between software worker and manager. While the software workers are more diffuse than they were in the 1970s and 1980s, they are not as diffuse as they were in 2000, prior to the dotcom bust, despite the NASDAQ Composite Index showing rapid growth in technology-oriented firms.
4.5.4 Labor practices and firm strategies

The labor practices and firm organization of the 1990s generally fit the typology of global, diversified firms that use both centralization and globalization as strategies to manage labor costs and extract value. The unpredictability of product cycles meant that firms resorted to temporary, contract, labor the most routinized tasks of programming including testing and coding. Firms also sought to manage their labor expenses by outsourcing particular tasks to specialized software programming firms or by offshoring work in countries with cheaper labor. The public perceived work in software to be glamorous and extravagant, like that of the work of video game work during the beginning of the 1980s.

Table 4.7: Diffusion of software labor across industries, 1970–2015.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Industries over 2.53% software workers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Industries over 7.19% software workers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


During the 1990s and 2000s, labor and firm strategies were archetypically flexible. The dot-com bust forced firms to reconfigure, focusing on managing labor costs associated
with the rampant growth and acquisition of software workers during the late 1990s and early 2000s. In the 1990s, large firms, especially Microsoft, began hiring firms to supply laborers to work with full-time employees on their campuses. These contracts were short 3- to 9-month contracts. Contractors felt like they were not being compensated fairly with regards to stocks and benefits. Microsoft contractors sued as I’ll explain in the next chapter. The dotcom bust reshaped the business use-case for software squarely in method to increased productivity, even those producing software, and firms shed their many of their software workers in pursuit of different methods.

4.5.5 Emergence of the cloud and mobile devices

The mass-market software period saw the miniaturization of computing and the popularization of smaller computing platforms like the tablet, the smartphone, and the laptop. Microsoft, Apple, and Linux platforms battled for dominance in desktop computers – in the personal and business operating systems and desktop software markets. Large software firms continued their pursuit of scientific management, but also relied on flexible contractors and peripheralized locations of production. Software production continued to concentrate. These characteristics match our expectation of a global software capitalist-variant.

Yet, software makers could no longer rely on mass-market software to produce the wealth they had previously enjoyed from selling boxed software commodities. Firms instead streamlined desktop software production, and pursued new avenues of profit. The emerging profit cycle depends on the growth of the internet, the required servers in service of the internet, and the servers to provide ecommerce and customer relationship management to clients. Microsoft, Apple, and Linux also fought for dominance on these computers which were not geared toward users per se but instead toward the automation
of routine services. In the mid and late 2000s, new cloud and mobile firms began to emerge, and existing companies began to restructure.

4.6 Late 2000s: Cloud, Mobile, Big Data

The current period of software development is difficult to characterize because of the newness of the relationships. Each of the three previous periods of software development links to a capitalist variant from Table 4.2 that characterizes technology, labor practices, firm strategies, their spatial relationships, and their capital-relationships. What separates this current moment, emerging sometime in the mid 2000s, is the entrenchment of global capital amidst new technical possibilities. Physically, larger clusters of computers with immense computing power in specially built data centers provide by firms with the storage and power to analyze large amounts of data. Paired with processors in internet capable devices like physical fitness trackers, watches, virtual-reality headsets, phones, smart city sensors, etc., a dual need emerges to build both large-scale and small-scale software. Machine learning, big data, internet-of-things (IoT) in the home or the city, and augmented reality, are some of the buzzwords used in industry. They are all interdependent on a set of technical components – large storage capacity for data, networked devices and servers, and the computing power for processing machine-learning algorithms – used to glean efficiencies in existing markets and open up new markets. Software and software workers demand the hardware capability to provide software instructions that drive these technological developments. These technological developments then guide, in part, the future structure of software development.

4.6.1 The Great Recession

Like the previous period, the this current period had a bubble of its own. The Great Recession was not a dot-com bust but instead a real-estate bust that drove the economy
into a slowing recession after its modest recovery after the dot-com bust. The real estate bust started in 2007 and caused what scholars call the “Great Recession” which caused consumers to spend less, causing firms to once again, justify the labor of each of their workers. Work in software continued to grow, just at a smaller pace, since managers saw software workers as providing efficiency to firms. The recession hit the finance industry hard, but the number of software workers grew during the Great Recession. While other industries shed other jobs, software workers remained. Fewer startups entered the scene during the Great Recession, and there is a gap in IPOs during this time.

Between the 2005–2009 (which comprises puts the start of the Great Recession squarely in the middle) and 2011–2015, software work was back on track and continued to grow. The economy added almost 500,000 additional software jobs, see Table 4.4, and software continued to gain comparative labor share of industries, see Table 4.5, and continue to diffuse into across all industries, see Table 4.1.

4.6.2 New technologies, faster internet

The proliferation of computing devices and the dichotomous relationship between centralized servers and consumer oriented devices relies on fast cellular data networks and faster wired connections providing near ubiquitous wifi access. Apple and AT&T released the first phone with a cellular data in 2007. Previous iterations of smartphones, like the Microsoft phones which ran Windows CE, relied on constant connections to wifi. Google released a competing phone in the next year and Microsoft soon after.

The distribution of handheld connected devices is only part of the story. New forms of interaction, like virtual reality, augmented reality, and touch screens, necessitated firms re-imagine how users interact with devices. Phones, tablets, VR headsets, glasses, and watches, provided new opportunities for software makers to port existing desktop software to a new era of devices, or imagine entirely new software.
The connected nature of the devices and the software that run on them meant that new forms of social interaction also made possible. “Social media” become a term to describe networks of connected peers. Social media software could run on distributed devices, however, they require immense centralized software to act as a repository for storing links between people and the content they share.

Two important related markets came with social media and connected devices that are important for this study. First, innovations in social media were also leveraged in software production to improve team efficiency. New firms used best practices from social media to curate software interactions and software tracking for both the production of software and for any other production-line or production process with complex steps and multiple production steps. Second, the immense amount of data collected provided new opportunities to segment customers and create targeted advertising on distributed devices. With regular users and a large enough user base, social media firms became advertising firms, controlling not just how users interact, but selling advertising space based on profiles and segments from data that users freely provide to the social media firms.

### 4.6.3 New firms, new markets

At the time of writing, firms like Microsoft and Apple dominant the software landscape. This is an incomplete list, as new firms dominate the the cloud and mobile sector in ways that would be unimaginable in the previous period. In the previous period, software as a commodity on every desk meant that the computer was the dominant hardware architecture for software commodities. The war over the desktop operating system continues to be waged. As processors (nano or otherwise) infiltrate existing devices like coffee machines, cabs, routers, light posts, TVs, and phones, or are core to new devices like virtual reality headsets, tablets, or e-readers, new markets for software emerge. And
importantly, the software on these devices must talk with each other or a large centralized server. New wars are waged for both cloud computing devices, smart phone devices, and IoT devices, while new firms sell software as a commodity for new platforms on servers, as smart phone apps, or smart-city software.

Firms like Amazon (founded 1994), Google (1998), Facebook (2004), Twitter (2006), and Snapchat (2011) are some of the largest players that barely, if at all, existed in the previous software period. Each has had an IPO that made the front page of the New York Times and each has a listing on the NASDAQ index. Amazon IPOd in 1997, the earliest of this group, but was still considered an online book store, not the cloud platform, advertising algorithm, grocery delivering, drone distributed, e-reader making, behemoth that we know today. Google, always a software company, had its IPO in 2004 while Snapchat, the most recent to IPO, did so in 2017. Microsoft, long a software commodity firm, had a public change of direction with the new CEO in 2014: Microsoft is now a cloud and mobile first software company. Each of these firms relies on the dichotomous technologies of large centralized servers run by the company and distributed computing devices for their clients. Microsoft, Amazon, and others, even lease out some of their own centralized server processing power and memory.

4.6.4 Labor processes and firm organization

These labor-market dominating firms centralize software design and production either in Silicon Valley, Seattle, or both, with localization (translation) work performed in peripheral locations outside the US. Google, a stereotypical Silicon Valley firm, has a location outside Seattle where it centralizes its Cloud Platform and Search Ads to leverage the talent pool of Microsoft and Amazon employees with similar experience.\(^6\) Apple, another stereotypical Silicon Valley firm, has a location in Austin where it has a support

\(^6\)Seattle product manager at a large firm. Interview, 10/01/2016, D23, transcript.
call center and a chip design lab. Apple’s location of the chip design lab is strategic:
Austin has a long history of chip designers and the salary requirements are much lower in
Austin.\textsuperscript{7} The support center is tangential.

As software becomes more common parts of everyday work, software workers in
other firms temper the dominance of these firms and their locational strongholds. These
firms make software that is a commodity yet as computers continue to employed in
production, firms employ software workers to integrate existing software platforms
and databases. Labor is still segmented, and retracting from its peripheralization of the
previous period, while simultaneously undergoing an automation as firms seek to use
new technologies to replace or reduce their need on software developers. One product
manager relayed that:

My job is to make sure my developers are programming as much as possible.
We have this contract, that if we do stand-up [daily 15 minute meeting] I’ll
stay out of their way the rest of the day – they want that too.

Later when discussing ideal candidates for a developer position, the interviewee laid out
expectations of skills, but also sets of software methods that they would bring to the
position.

I expect my developers to use develop tools, scripts, methods, that take some
of the day-to-day programming out of the way – make them more efficient at
what they do.\textsuperscript{8}

Other interview subjects in other locations, even some with experience in Silicon Valley,
corroborated this account of the relationship between manager and worker. Managers
keeping their workers on task is hardly a new phenomenon. Programmers, however,

\textsuperscript{7}Austin developer and entrepreneur. Interview, 11/02/2015, D11, transcript.
\textsuperscript{8}Seattle product manager at a large firm. Interview, 10/01/2016, D23, transcript.
asking their managers to skip meetings and seeking their own tools for efficiency sits contrary to the antagonist shop-floor relationship between worker and manager that is commonly told in these settings. Programmers also realize the role their managers play in guiding their career, discussed in the currency of “innovation” by one interviewee.

If you don’t have a good manager, you’re not going to innovate. Doesn’t matter which side [manager or worker] you’re on. The thing about innovation is that it a lot of times goes counter to making money now. It’s making money now versus making money later. If you don’t have good management that will say, look, we have to not make money now so that we can make money later... You have to have management to have the freedom, and the intellect to make that call.⁹

Among smaller firms, if pay is lower and software is not the primary purpose of the firm, then the antagonistic relationship is more likely to persist. Interview subjects I interviewed, in line with the literature, accept lower pay and antagonistic relationships with managers if they can gain experience to move to other firms.

The relationship to managers is just one of the relationships the above quotes highlight. The other is the relationship to technologies, specifically, the use of using code to write code. This generally comes in the forms of libraries and higher languages. The networked nature and inter-operable ideals of the fourth period allow for software workers to leverage APIs (Application Program Interface) so that software programs can interact across computers even if they run different hardware architectures, run different operating systems, run different software, and are physically separated. Firms like Amazon, Google, and Microsoft leverage this set of relationships to provide cloud infrastructure and cloud services. Microsoft licenses access to a set of machine-learning APIs so that software

⁹Austin developer at an SME. Interview, 11/03/2015, D13, transcript.
workers can integrate machine-learning algorithms into desktops, laptops, and mobile devices that do not have the computing power to themselves run the computationally heavy machine-learning programs.

Software workers can use this cloud infrastructure, cloud services, and open machine-learning libraries, like Google’s TensorFlow – a fifth generation language used in combination with python for machine-learning – to aid in software production. Software workers use this set of technologies for a wide variety of tasks. One interviewee corroborated the use of a software product internally within their firm that used machine-learning to assign programming tasks to the software workers on their team based on their productivity, skills, and deadlines. During a code sprint planning, when software workers and their managers assess software goals for the next two- to four-week period, then divide their desired output into divisible tasks, they use the machine-learning algorithm to provide initial assignments. During the code sprint, the software alerts the entire team if the team is not projected to complete the code sprint on-time and the team will then either revise their goals, work harder, or hire a temporary contractor. This is the same interviewee that was explaining important aspects of a good manager and later stated that his current manager was the best manager he had. In context of the manager above which stated that his job was to make sure that his developers were always coding, this logic is obvious: as soon as a software worker completes their assigned and verifies its functionality, the machine-learning software will assign a new tasks based on that particular workers skills and the the rest of the team. In one sense, even, one might carry the argument further and argue that this application desskills the manager.

Two of the most popular tasks for machine-learning are for image classification and behavior modeling; core components for providing driving instructions to self-driving

---

10 Code sprints vary in length, based on the decisions made internally within the team, but this is a typical length.

11 Austin developer at an SME. Interview, 11/03/2015, D13, transcript.
cars and, more importantly, self-driving freight trucks. These vehicles take thousands of pictures per minute, send them to an image classification system that identifies its surroundings, speeds of other cars, and curves in the road, then passes these to a behavior modeler to determine the next course of action. This *specific* application, again, deskills those who sell their labor to drive. The *broad* development of machine-learning deskills software workers that were using other methods because of the increased productivity of software workers – measured by the time needed to develop a set of hardware instructions – using machine-learning. When software workers use machine-learning, or other forms of software automation, they don’t perceive their actions as automating their own work, or even other software workers, but instead as providing new capabilities. These new capabilities are the process which deskill other software workers, requiring that software workers reskill themselves with the latest library, technology, or high-level language. I continue this discussion in Chapter 6 to show how certain types of software occupations are more vulnerable to these processes of deskilling.

4.7 Conclusion

This chapter broadly sought to understand the nature and extent of software production activity since the formal development of the software occupation in 1970. I asked three specific questions:

1. What is the extent of software production activity in the US?

2. In what ways has software production activity changed in the US since 1970?

3. What is the relationship between software production activity and larger circuits of production?
To understand the extent of software production activity in the US, I used occupation and industry data from the US Census and the ACS across six periods between 1970 and 2015 provided by the IPUMS USA. This chapter has shown that software production is best understood as labor in the extension of technical management that spreads across productive activities. That is, software production, like management or clerical work, is best understood as a laboring activity that is integral in new and established production processes of industries ranging from health care, food production, biotechnology, finance, education, extractive industries, and so on. I show the results of this work in Table 4.7 and Figure 4.3. Software workers are in every industry as of 2011–2015 data, and have been since 2000. Further, since 2000, there are over 12 industries were software occupations make up significant portions of the industry’s labor force. I also showed that software workers were 0.35% in 1970 and that the 4.3 million software workers in 2011–2015 comprise 2.66% of the labor force. By showing that software workers produce software across industries as a integral component of capitalism, I demonstrated that scholars must reject the ease of identifying one or more software “industries” or sectors to understand software production. Drawing boundaries around software production activity by using “industry” as a unit of analysis excludes software production in other industries. Further, an industry definition would miss prolific examples of firms like Amazon and Uber. Is Amazon a wholesale distributor or the dominant online US retailer and cloud infrastructure provider? Is Uber a car-sharing service or a software app? One would miss the vast array of software workers inside these firms using an industry definition.

To determine the ways software production has changed in the US since 1970, I return to the IPUMS USA data and combine it with interview data and historical research. Through interview and historical research I extend and elaborate on a framework of software periods in Table 4.1. Further, I showed how a fourth period of software is in the
making. Interview data confirmed these periods and informed the current, fourth, period. One veteran developer in Portland, formerly of Seattle, put the last three periods in his own words.

There’s really been three moments of software. The transition, in the 1980s, from scratch programming to object-oriented programming [1970s and 1980s, object-oriented emerged in 1972 out of Xeroc PARC]. The transition from installed to web-based software – you know from burning CDs and floppy disks to software housed on the internet ... and the movement to the mobile and cloud, even before the iPhone, maybe with Windows CE [1990s and early-2000s, web gains popularity in early 1990, Windows CE introduced 1996]. ... Then ... there’s three big things, technologies or platforms, right now that could really drive the next moment, they’re all emerging, and have been ‘emerging’ for a long time: IoT, machine learning, and augmented reality [late-2000s onward].

This chapter showed that the fourth period draws on the connected nature of personal devices, web/mobile apps, and cloud computing. I show that each of the software periods represent the profit cycles of firms that heavily rely on software during that time, some of which are able to reconfigure themselves across periods – like IBM’s transition between the contractor period and the corporate software period but decline in the mass-market period, or Microsoft’s transition between the mass-market period and the cloud/mobile period. Interview data and historical research document the changing software technologies, language generations, and automation across periods. I detailed the workplace changes across each period along with changing nature of labor dynamics, noting the capital

---

12Portland veteran developer. Interview, 10/01/2016, D19, transcript.
relationship go from machinofacture to post-Foridst with evidence of computational flexibility.

From Table 4.4, I show that between 1980–1990 and 1990-2000 a significant growth period for software production activity provides the labor force for the diffusion of software production activity and mass-market software. More specifically, that 1990 and 2000 are the only observations that the number of software laborers more than doubled. Using a modified coefficient of localization, I showed that 2000 had the highest rate of software worker diffusion across industries, and that the dotcom bust in 2000 had lasting impacts on software workers' industry composition. Despite an overall trend toward diffusion in 4.7, after 2000, software workers began to concentrate in information, insurance, and business based industries. Primary and secondary interview data show the persistent attempt by management to rationalize the labor of software workers, but new software technologies deskill software workers and render management rationalizations out of date.

To address the final question, the relationship between software production activity and larger circuits of production, I introduce capitalist variants and industrial districts to the discussion of software periods and profit cycles. Figure 4.1 documents this relationship.

With the introduction of capitalist variants, I argue that software production’s four periods have a direct relationship to capital’s historical variants. I show that two characteristics of software enable the microcosm of software production activity within capitalism to mimic these larger structural shifts in a short amount of time. First, the production of software requires existing software, facilitating the rapid historical trajectory of what software is available and what values become codified in the software. Second, the capacity for a capitalist to change the functioning of machinery in pursuit of profit increases because the capitalist no longer must build a new physical machine. The “low-materiality” of
software allows the capitalist to reconfigure the digital machine by changing the software. With the right knowledge, changing a few lines of code can significantly reconfigure the digital machine. Changing the entire codebase can reconfigure the digital machine in totality. The low-materiality and ease of reconfiguring software increases competition between capitalists because of a heightened sensitivity between profit computing time, accelerating the need to develop new software efficiencies and innovations. Competitive pressures and faster rates of software development accelerates the falling rate of profit, creating additional competitive pressures, requiring faster software, and accelerating the flow capital along capital circuits. I conclude, then, that if software rapidly developed along historical variants of capitalist production, then software’s emerging relationships may provide insights into the future forms of capital production outside of software. The form and changes within software production activities may be prototypical for other types of production and its respective restructuring, geographies, and divisions of labor.

With the introduction of industrial districts, I argue that each of these periods has specific profit cycles and industrial district formations. While I introduce broad geographical shifts in this chapter, the next chapter analyzes the regional character of software production. I then examine three industrial districts to contribute back to the discussion on of the role of industrial districts in software periods, profit cycles, and capitalist variants.
**5 Variegated Technopoles, Production, Wages**

Rapidly growing regions exhibit distinct varieties of industrial district structure.

Gray, Golob, and Markusen 1996, 651

Contrary to common wisdom, high technology varies dramatically from place to place.

Cortright and Mayer 2001, 2

If each local labor market represents a unique geographic conjuncture of labor market processes, it follows that the institutional form of the wage-employment relationship ... also will take on a locally distinctive character.

Peck 1996, 110

**5.1 Introduction**

Software traces its genealogy through the places, industries, firms, and workers of semiconductor and computer manufacturing. Software operation, after all, relies on the semiconductors in computers to operate. The first high-level software languages separated the skills needed for software production from that of hardware design and engineering. As Kraft put it, “high-level languages make it unnecessary for programmers to be quasi [hardware] engineers” (1979, 146). This initial separation is the foundation for new
forms of production specific to software. That is, software production began to require
the use of software for software development, testing, and deployment – requiring new
skills, inputs, firm structures, and spatial arrangements. Programmers, along with other
software occupations that I document in this chapter, began an uneasy relationship both
producing software and relying on it for production.

The history of software places its origins in regions like New York, Route 128, and Sil-
icon Valley. As both computing hardware and software production matured and separated,
software production became less dependent on the regions with hardware production
agglomerations – whether concentration of semiconductor chip design or computer man-
ufacturing more broadly. While software and hardware remain inextricably linked, new
high-level languages continue to abstract software development away from hardware and
even lower-level languages. The continued weakening of the interdependency between
software production and hardware production suggests a continued weakening of the
regional co-location of software production and hardware production.

This chapter seeks to understand the regions of software production. Addressing the
changes above, this chapter asks:

1. How does software production activity vary between region?

2. What are the industrial district formations of software production in Seattle, Wash-
ington, Portland, Oregon, and Austin, Texas?

3. What are the regional outcomes, via occupational structures and income distribu-
tion of software workers, of regionally specific modes of software production?

This chapter builds on the idea that second-tier cities, as dubbed by Markusen, DiGio-
vanna, and Y. S. Lee (1999), play important roles in regional economic development and
in shaping the global circuits of production. Silicon Valley, comprised of the metropolitan
areas of San Francisco and San Jose, has long been the dominant region for software production. Software production in second-tier cities – like Seattle, Portland, and Austin – continually challenge and compete with Silicon Valley for dominance. Between each software period, second-tier cities (regions) provide locational windows (Storper and Walker 1989b), new regional locations that may unseat Silicon Valley’s dominance.

Software production’s low-materiality affords rapid cycles of development. Thus, as firms and workers in second-tier regions develop new software, new forms of regionally specific software development emerge. This chapter argues that second-tier regions have regionally specific modes of production and that these regions may exhibit differences along forces of production (technologies, skill-sets, labor markets), social relations (class-based firm-worker relationships), and/or spatial organizations (industrial district formations).

In this chapter, I trace the shifting geography of software production across US metropolitan areas (metros) since 1970. I use interview data to interrogate the relationships between regional outcomes and the technologies and production methods used in three specific second-tier regions: Seattle, Washington; Portland, Oregon; and Austin, Texas. Then, combining interview data and software occupation data, I construct industrial district types to understand the three regions’ industrial structures, relationships with other regional software agglomerations, and relationships with one another.

First, in the section that follows, I document regional software agglomerations since 1970. In the second section, I combine this historical context and interview data to compare and contrast the industrial districts and structures of Seattle, Portland, and Austin. In the last section, I show how differences in industrial structures yield different regional occupational structures and income distributions.
5.2 Regional Differentiation of Software Occupations

In 4, I showed how software production activity spans multiple industries and is, therefore, best measured through software production labor. To identify and differentiate *regional* software agglomerations, this section measures software production labor in US metros since 1970. Two methodological issues have hindered research in this area. First, researchers must use boundaries to define metro regions, but government agencies change these definitions over time. The US Census Bureau defines metro by grouping counties together, and while county boundaries are relatively stable, metros tend to absorb counties as the region grows and commute patterns in and around the area change. Second, scholars need access to US Census Bureau data where cross-tabulations are possible, and the only source offering this flexibility – the Integrated Public Use Microdata Series (IPUMS) – comes at the cost of geographical comparability. IPUMS cross-tabulations allow scholars to determine, for example, the number of software laborers and the laborers’ race/ethnicity, while other sources limit researchers to either the number of software laborers or the laborers’ race/ethnicity. The geographies provided by the IPUMS are in special geographies called Public Use Microdata Areas (PUMAs) for 1990 and later and county groups for 1970 and 1980. The geography types are not only inconsistent, but the geographies themselves change with each Census and do not nest with the US Census Bureau definitions of metropolitan areas.

Appendix A documents a novel approach to provide consistent cross-tabulations at the metro level since 1970. At a high level, this method assigns a portion of a PUMA geography’s population to its overlapping metro area according to the US Census Bureau’s 2013 definitions. For each 10-year Census period and PUMA geography period, the proportion of PUMA’s population living within the metro area determines the portion of a PUMA’s population assigned to its corresponding metro area. In this calculation, the

I leverage this method to calculate the largest metro agglomerations of software workers for the six time periods noted above. Interested in large software agglomerations, I present the largest 25 in Table 5.1. Seeking to narrow the scope, I then identify regions that have large shares of software workers. Table 5.2 shows the 15 largest metro agglomerations of software workers by share of the metro labor force over the same six time periods. Both tables contain summary statistics for all 51 large metros and the entire US. The three cases selected in this dissertation – Seattle, Portland, and Austin – are listed in bold in with rankings outside of the top 22 or 15 noted at the bottom of the respective table. Arrows in Table 5.2 document significant changes among case regions or their change in respective ranking.

Tracking the number of software workers in Table 5.1 during the 1970s appears almost unremarkable. The largest 8 metropolitan areas take up the first 8 largest regional software

Table 5.1: Metro software agglomerations by year, number of software workers, 1970–2015.

<table>
<thead>
<tr>
<th>Metro</th>
<th>1970 000s</th>
<th>Metro</th>
<th>1980 000s</th>
<th>Metro</th>
<th>1990 000s</th>
<th>Metro</th>
<th>2000 000s</th>
<th>Metro</th>
<th>2009 000s</th>
<th>Metro</th>
<th>2015 000s</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYC</td>
<td>39.0</td>
<td>LA</td>
<td>22.7</td>
<td>Chicago</td>
<td>16.9</td>
<td>Philadelphia</td>
<td>10.5</td>
<td>Boston</td>
<td>9.9</td>
<td>DC 9.6</td>
<td>SF 8.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>39.0</td>
<td>LA</td>
<td>32.2</td>
<td>Chicago</td>
<td>26.5</td>
<td>DC</td>
<td>22.4</td>
<td>Boston</td>
<td>19.2</td>
<td>Philadelphia</td>
<td>18.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NYC</td>
<td>286.8</td>
<td>LA</td>
<td>155.2</td>
<td>Chicago</td>
<td>140.4</td>
<td>LA</td>
<td>133.0</td>
<td>SF</td>
<td>113.7</td>
<td>Dallas</td>
<td>111.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>14.4</td>
<td>Philadelphia</td>
<td>39.0</td>
<td>DC</td>
<td>60.3</td>
<td>LA</td>
<td>140.2</td>
<td>Chicago</td>
<td>155.2</td>
<td>Minneapolis</td>
<td>12.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>43.9</td>
<td>Portland</td>
<td>36.5</td>
<td>Portland</td>
<td>43.4</td>
<td>Austin</td>
<td>32.8</td>
<td>Columbus</td>
<td>35.7</td>
<td>Columbus</td>
<td>33.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Selected metros not in the top 25 when ranking among metros over 1 million persons:

<table>
<thead>
<tr>
<th>Portland</th>
<th>1.0</th>
<th>28 Portland</th>
<th>3.7</th>
<th>27 Portland</th>
<th>8.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austin</td>
<td>0.5</td>
<td>30 Austin</td>
<td>3.2</td>
<td>28 Austin</td>
<td>8.7</td>
</tr>
</tbody>
</table>

Large metros (n = 51)

- median: 2.0
- total: 218.2
- National: 291.0
Table 5.2: Metro software agglomerations by year, share of labor force, 1970–2015.

<table>
<thead>
<tr>
<th>Metro</th>
<th>1970 000s</th>
<th>%</th>
<th>1980 000s</th>
<th>%</th>
<th>1990 000s</th>
<th>%</th>
<th>2000 000s</th>
<th>%</th>
<th>2009 000s</th>
<th>%</th>
<th>2015 000s</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Jose</td>
<td>6.0</td>
<td>0.79</td>
<td>13.0</td>
<td>1.29</td>
<td>31.8</td>
<td>2.65</td>
<td>81.3</td>
<td>6.27</td>
<td>80.2</td>
<td>6.07</td>
<td>102.6</td>
<td>7.21</td>
</tr>
<tr>
<td>Rochester</td>
<td>3.8</td>
<td>0.46</td>
<td>7.6</td>
<td>0.97</td>
<td>8.5</td>
<td>2.03</td>
<td>27.7</td>
<td>4.46</td>
<td>173.0</td>
<td>4.09</td>
<td>Seattle</td>
<td>119.9</td>
</tr>
<tr>
<td>MSP</td>
<td>5.9</td>
<td>0.45</td>
<td>22.4</td>
<td>0.82</td>
<td>60.3</td>
<td>1.81</td>
<td>155.2</td>
<td>4.12</td>
<td>DC</td>
<td>94.6</td>
<td>4.25</td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>9.6</td>
<td>0.44</td>
<td>13.0</td>
<td>0.80</td>
<td>14.1</td>
<td>1.65</td>
<td>SF</td>
<td>113.7</td>
<td>Raleigh</td>
<td>30.5</td>
<td>3.74</td>
<td>DC</td>
</tr>
<tr>
<td>LA</td>
<td>22.7</td>
<td>0.39</td>
<td>19.2</td>
<td>0.69</td>
<td>45.1</td>
<td>1.47</td>
<td>Austin</td>
<td>33.8</td>
<td>Austin</td>
<td>110.1</td>
<td>3.59</td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>8.2</td>
<td>0.39</td>
<td>12.4</td>
<td>0.67</td>
<td>38.9</td>
<td>1.43</td>
<td>Boston</td>
<td>18.7</td>
<td>SF</td>
<td>108.6</td>
<td>3.39</td>
<td></td>
</tr>
<tr>
<td>Boston</td>
<td>9.9</td>
<td>0.39</td>
<td>14.4</td>
<td>0.61</td>
<td>40.8</td>
<td>1.28</td>
<td>DC</td>
<td>55.4</td>
<td>3.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hartford</td>
<td>3.0</td>
<td>0.39</td>
<td>9.9</td>
<td>0.64</td>
<td>25.2</td>
<td>1.30</td>
<td>Seattle</td>
<td>72.1</td>
<td>Dell</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>NYC</td>
<td>39.0</td>
<td>0.35</td>
<td>14.4</td>
<td>0.61</td>
<td>40.8</td>
<td>1.28</td>
<td>MSP</td>
<td>67.1</td>
<td>Denver</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Portland</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Portland</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>4.3</td>
<td>0.34</td>
<td>36.4</td>
<td>0.55</td>
<td>15.7</td>
<td>1.19</td>
<td>Austin</td>
<td>87.7</td>
<td>Austin</td>
<td>55.5</td>
<td>2.89</td>
<td></td>
</tr>
</tbody>
</table>

Selected metros not in the top 15 when ranking among metros over 1 million persons:
- Austin
- PDX

Large metros (n = 51)
- mean 0.25 0.42 0.90 2.11 2.12 2.36
- median 2.0 0.25 4.0 0.37 9.8 0.80 27.7 1.83 30.5 1.88 40.2 2.22
- total 218.2 0.30 397.7 0.46 947.7 0.97 2419.4 2.23 2620.2 2.19 3095.3 2.44

National 291.0 0.21 540.0 0.33 1289.0 0.71 3294.4 1.65 3618.3 1.68 4245.0 1.92

Selected metros not in the top 15 when ranking among metros over 1 million persons:
- Austin
- PDX

Large metros (n = 51)
- mean 0.25 0.42 0.90 2.11 2.12 2.36
- median 2.0 0.25 4.0 0.37 9.8 0.80 27.7 1.83 30.5 1.88 40.2 2.22
- total 218.2 0.30 397.7 0.46 947.7 0.97 2419.4 2.23 2620.2 2.19 3095.3 2.44

National 291.0 0.21 540.0 0.33 1289.0 0.71 3294.4 1.65 3618.3 1.68 4245.0 1.92

agglomerations, almost in exact order, Detroit having the lowest number of software workers of this group, despite being the fifth largest US metro in 1970. With a couple exceptions, the list of software agglomerations corresponds approximately to metro region size. San Jose’s appearance after Detroit, in 1970, however, suggests that, despite the region’s metro population of around 1 million, the city already had the beginnings of a software agglomeration. Notably, Rochester and Hartford appear on the list despite having small metro populations. The list places Portland and Austin among the smallest software agglomerations, having 1,500 software workers between the two regions.

With Boston, San Francisco, and San Jose – the subjects of work by Saxenian (1996) – remaining consistent, one might overlook a small number of important changes between 1970 and 1990. The growth of software occupations in Washington DC catapults the region behind only Los Angeles and New York in 1970 and behind only New York from 2000 onward. Sun Belt and West Coast regions, like Phoenix, Seattle, and San Diego, all rise in relative rankings. Regions like Philadelphia, Detroit, and Rochester all fall in relative rankings. Hartford rises in relative rankings, then falls and drops off the list by 2000. While some of these changes can match to larger, national changes in population and migration dynamics, military and government contracting play a role, giving Dallas, DC, and San Jose significant positional advantage over their peers in the 1950s and 1960s. IBM’s dominance in hardware and software during the 1950s and 1960s also propels Hartford and Rochester on this list during the 1970s, 1980s, and 1990s. Rochester makes
its only appearance on this list in 1980, and Hartford makes its last appearance in 2000—coinciding with the decline of Bell Laboratories, the decline of Xerox, and IBM’s decision to outsource its operating system to Microsoft.

Minneapolis–St. Paul (Minneapolis) and Houston remain in the relative middle of the list. Both regions had significant computer manufacturing and became anchor-organizations as the region shifted toward software production. In Minneapolis, Honeywell Computers was an anchor-firm, and IBM opened its Rochester, Minnesota (not Rochester, New York as listed in the table) production facility which partnered with the University of Minnesota to develop IBM’s Blue Gene, once the world’s fastest supercomputer. When Honeywell and IBM declined in the region, Minneapolis’s hardware and software workers, with help from a new local technology coalition, partnered with the medical industry and academic research and development (Slaughter and Rhoades 1996). This allowed Minneapolis to maintain its software agglomeration. Similarly, Compaq started in Houston, and was perhaps the most successful direct competitor to IBM. Because of their similar Intel architecture and software compatibility, Compaq computers were able to emulate IBM computers and run software that had been designed for IBM computers. Compaq’s growth over the next two decades led to its acquisition of the Boston-based Digital Equipment Corporation (DEC). Compaq/DEC had, at its height, over 140,000 employees worldwide. Silicon Valley-based Hewlett-Packard purchased Compaq/DEC in 2002, shifting production to its own facilities and retiring the brand. Throughout the 1980s, Houston leaders and the Texas Division of Economic Development, sought industrial diversification to avoid Houston’s reliance on oil. Through public and private funding, Texas government, Houston government, and local elites poured money into attracting external firms and expanding local research in energy, biotechnology, space enterprises, and international business services. The investment made in Houston – when its software agglomeration was healthy – allowed the city to retain its
software workers as they worked in other industries (Feagin 1988).

Houston and Minneapolis software workers were able to move from hardware industries, as the computing and semiconductor industries consolidated and restructured, to other complementary industries after 1990. Even Boston’s status on this list – maintaining a steady fifth place from 1970 to 1990, then a steady seventh/eighth place for the period 2000-2015 – suggests a similar story despite the story of Route 128 from Saxenian (1983). As early anchor-firms of Route 128 declined and lost out to San Jose and San Francisco, other institutions, like Massachusetts Institute of Technology, supplemented then replaced the role of anchor-organization. Hartford and Rochester software workers were not able to apply their skill-sets to other local industries and no anchor-organization provided software jobs to replace those lost through the regional downsizing of Bell Laboratories, Xerox, and IBM (Stanback 1985). The result for these two regions was a loss of relative growth and ranking in software agglomerations.

Meanwhile, an emerging biotechnology industry and military/government contracting allowed DC to move up the list (Chapple et al. 2004; Cortright and Mayer 2001). Similarly, Dallas, benefited from military and government contracting through its anchor-firm, Texas Instruments. The software agglomeration in Dallas grew tenfold between 1970 and 1990, more than any other metro on this list. Dallas continues to maintain a strong software agglomeration, and Texas Instruments is still an anchor-firm in the region (Stanback 1985). After 1990, other industries, like finance and insurance, employed large numbers of software workers in the Dallas region.

Comparing the list of software agglomerations in 1970 to the list of software agglomerations in 2015, three observations are worth noting. First, software production activity in 1970 appears evenly distributed across US metro regions with the exceptions noted above. Second, regions with military and government contracts or those able to transition software workers from one anchor-organization to another maintained or grew their
software agglomerations relative to other metro regions. Third, the three time periods starting in 2000 mark a distinct break from the previous three periods, documented in Table 4.5 and Figure 4.3, as software spreads into other industries – and ultimately new second-tier regions – because of the success of mass-market software of the 1990s and 2000s.

The purpose of Table 5.2 is to show the primacy of the software agglomerations for large metro regions. The table reads as a typical list of software clusters across the US. Metros with military or government contracts or IBM presence like Minneapolis, Hartford, and Rochester dominate the 1970s and 1980s. By 2009 and 2015, Silicon Valley (although, San Jose has been at the top of the list for the entire 45-year span), Seattle, Research Triangle, and DC dominate the top of the list. These regional changes in software agglomerations coincide with the four software periods in Table 4.1. The next section will focus on the three case regions of this research, discuss their rise in both the number of software workers and regional share of software workers, and interrogate their modes of production.

5.3 Constructing an Industrial District Typology

In seeking to answer how software production activities vary between regions, the previous section documented the varying levels of regional software activity and their changes over time. This chapter seeks to better understand the rise, and relative success, of three specific regions and their modes of production through interviews with workers, managers, and key informants, further interrogation of the IPUMS data, and participant-observation at conferences in each region. I will then compare the inner-workings of each region’s software production activity to its idealized industrial district typology from Markusen (1996). I extend and improve on this stylized industrial district typology in 5.3 by contributing software-specific variances based on Campbell-Kelly (2004), S. D. Graham
The rest of this section will use the mixed-methods research mentioned above to understand the software agglomerations of three regions: Seattle, Washington; Portland, Oregon; and Austin, Texas. The stylized software districts presented in Table 5.3 are just that: stylized. The types are hypothetical district configurations and each of the case regions does not necessarily map onto each region. I note where there are deviations and seek to understand them. As a framework, they provide useful starting points for understanding the dynamics of regional modes of software production and their regional impacts. Markusen (1996) is quick to remind us that no region is a perfect match to an industrial district type, and sometimes, as I’ll argue about Austin, a region may be a mixture of two or more types – or may be in transition to a new type. In fact, as we saw with Hartford in the previous section, the persistence of a region’s district may be in jeopardy altogether.

5.3.1 Seattle, Washington: a classic hub-and-spoke software district

The Seattle region represents a classic hub-and-spoke industrial district structure. In her book about industrial districts, Markusen (1996) uses Seattle as an example of a hub-and-spoke region. The key characteristic for a hub-and-spoke region is that one or more large vertically integrated firm(s) dominates a sector’s production in the region. This anchor-organization, or the hub firm, dominates locally because of its stature in external markets on national and international levels.

In Seattle, for instance, the economy is organized around Weyerhauser as the dominant resource sector company, Boeing as the dominant industrial employer (commercial aircraft and military/spacecraft), Microsoft as the leading
Table 5.3: Technological characteristics of software industrial districts.

<table>
<thead>
<tr>
<th>Technological Dynamics</th>
<th>Hub-and-Spoke</th>
<th>Marshallian</th>
<th>Satellite-Industrial</th>
<th>State-Sponsored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>3G+, deployment/to-market efficient, strong specification, custom, platform compatibility</td>
<td>3G+, prototyping &amp; scalability efficient, flexible specification, interoperability</td>
<td>Legacy, early 3G languages, external</td>
<td>Mix of language generations, stable, strong specification, external</td>
</tr>
<tr>
<td>Innovation</td>
<td>Moderate-high, within firm, external acquisitions, capital-intensive</td>
<td>Moderate-high, within district via worker- or firm-relationships</td>
<td>Low, external, firm- or industry-based</td>
<td>Moderate, internal to project, capital-intensive</td>
</tr>
<tr>
<td>Automation</td>
<td>Development of automation for external firms, use of worker automation tools in development</td>
<td>Development of automation for firm and workers, use of automation tools in development</td>
<td>Use of external automation tools, production-oriented</td>
<td>Use of external automation tools, some development</td>
</tr>
</tbody>
</table>

services firm (software), the Hutchinson Cancer Center as the progenitor of a series of biotechnology firms, and the Port of Seattle as the transportation hub. (Markusen 1996, 32)

Seattle has several linked hub firms with global statures that span multiple industries, and the region benefits directly from the presence of research activity at the University of Washington. Boeing’s military orders for the Korean War and Cold War as well as the success of its 707 jetliner required recruitment of engineers – and the University of Washington’s expansion plan during the 1950s and 1960s aimed to address this demand (Abbott 1992). The University of Washington is where Bill Gates and Paul Allen tested software programs written by engineers in exchange for access to the institution’s mainframe computers.

Interestingly, the absence of successful semiconductor manufacturing differentiates Seattle from other well-known software regions. Instead, the presence of Boeing paired with the increasingly important technical aspects of aerospace may have given Seattle its initial, moderately-sized software agglomeration. Investment in UW paired with Boeing’s return to success spurred by the Vietnam War seeded Seattle’s software production. Today, the Seattle region is home to two software titan hubs: Microsoft and Amazon. Redmond is home to Microsoft’s headquarters, a suburban campus east of Seattle, and Seattle is home to Amazon’s campus. Both firms have offices in the region’s three main cities: Seattle, Bellevue, and Redmond.

Microsoft, founded in 1975, is the older of the two hub firms, but didn’t relocate to suburban Seattle until 1978. Microsoft grew steadily in Seattle selling its BASIC programming software, but began rapid growth in 1980 with its purchase of Seattle Computer Products’ Intel-based operating system SCP-OS, the re-branding of that operating system to MS-DOS, and the resale of MS-DOS to IBM for shipment with all IBM computers
Market exposure of MS-DOS, and the ability to sell MS-DOS to other computer manufacturers, helped co-create a steady computing platform alongside Intel processors and Intel-compatible processors. Microsoft, already with large market exposure, was able to innovate in operating systems and develop software packages that would run on MS-DOS and later Windows during the commodity period of software production. Microsoft’s success selling gaming platforms, and the platforms’ cross-compatibility with Windows games, required Microsoft to invest in cloud-computing capacity to enable online multi-player gaming.

Microsoft wasn’t alone in this endeavor; Seattle’s other titan hub, Amazon, would also find itself pursuing cloud-computing. Founded in 1994, Amazon had a brief stint in suburban Bellevue before moving to Seattle. After excelling in online book sales in its first couple years, Amazon went public in 1997 and began an aggressive expansion beyond books and into international markets a year later, becoming the dominant online retailer surpassed only by Alibaba in China. Amazon moved into search, competing with Google, Yahoo, and Microsoft in 2003, and later, in 2006, began offering cloud computing and online storage services. Both of these services were core components of Amazon’s online retail business: it needed to allow customers to search for products and it needed a robust real-time solution to track the site’s rapid rate of online transactions. Amazon’s cloud infrastructure was among the first offered as a product and rapidly became one of the most popular. Amazon later hosted Twitter, enabling Twitter to scale as its user base grew. Amazon’s most profitable segment is its web services, the umbrella for its entire cloud infrastructure. In the first quarter of 2016, Amazon’s profit from its web services was $604 million (Amazon.com 2016).

Today, both firms sell numerous other devices – including mobile phones, tablets, and gaming platforms – and have ventured into emerging markets like machine learning, augmented reality, and the smart home. Notably, both Microsoft and Amazon have
invested significant capital in their cloud-computing infrastructures because cloud computing was complementary to their core business at the time, but they now both resell cloud computing, cloud storage, and cloud-based machine-learning. The firms compete directly with each other but have both been able to continue to be profitable and grow since the market itself is still growing. In fact, Amazon’s cloud infrastructure has become its most profitable market, even amid its dominance as an online retailer. Importantly for Seattle software workers, cloud computing requires significant development and operations labor aside from the labor demands of the firms’ other business ventures.

Table 5.4: Industrial composition of software workers in the Seattle region, ACS 2011–2015.

<table>
<thead>
<tr>
<th>Industry</th>
<th>n</th>
<th>% of software labor</th>
<th>% of labor force</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer systems design and related services</td>
<td>49,946</td>
<td>41.65%</td>
<td>1.39%</td>
</tr>
<tr>
<td>Electronic shopping</td>
<td>4,957</td>
<td>4.13%</td>
<td>0.14%</td>
</tr>
<tr>
<td>Software publishing</td>
<td>3,827</td>
<td>3.19%</td>
<td>0.11%</td>
</tr>
<tr>
<td>Aircraft and parts</td>
<td>3,716</td>
<td>3.10%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Management, scientific and technical consulting services</td>
<td>3,510</td>
<td>2.93%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Aerospace products and parts</td>
<td>3,495</td>
<td>2.91%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Other telecommunication services</td>
<td>3,175</td>
<td>2.65%</td>
<td>0.09%</td>
</tr>
<tr>
<td>Insurance carriers and related activities</td>
<td>2,858</td>
<td>2.38%</td>
<td>0.08%</td>
</tr>
</tbody>
</table>

Total number of software workers in labor force 119,930
Total number of workers in labor force 1,977,709


The trajectory of Seattle’s software agglomeration is tied to Microsoft’s and Amazon’s capacities to innovate and expand into new markets. In both Table 5.1 and Table 5.2, there is not a single period where Seattle loses ground: relative to other metros, it has
grown both in raw numbers of software workers and in the share of software workers in the labor force relative to other metros. Microsoft was hit hard by the dotcom bust in the early 2000s. Windows remains the most popular operating system on desktop computers despite its share continuing to decline, but it is far from the most popular on mobile phones (Android) or server computers (Linux). While Microsoft’s growth slowed as it reconfigured, Amazon’s transition and diversification from online retailer to online services provider enabled the region to maintain its status and labor market. Seattle benefited from Microsoft’s and Amazon’s ability to innovate and expand from their original markets, and the coincidental timing of Microsoft’s lull and Amazon’s
 Silicon Valley firms seek access to Seattle’s labor market. Google produces cloud- and mobile-computing products in its large office in suburban Kirkland; Adobe produces its InDesign and AfterEffects software products in Fremont, near Amazon’s new Seattle headquarters; and T-Mobile’s US headquarters in Bellevue produces its mobile phone apps. These production facilities leverage the local worker expertise, but as satellite locations, have weak relationships with local hub firms.

Local governments unanimously support the hub firms and their complementary sectors. The municipalities of Seattle, Bellevue, and Redmond have worked independently with Microsoft, Amazon, and T-Mobile around infrastructure projects, transportation planning, and tax incentives. Seattle’s hub firms seek high economies of scale in production, maximizing the reach of their software or websites, focusing on mass-exposure. Small spoke firms in the region are dependent on the success of the hub firms, and there is little infrastructure for risk-sharing among software firms in the region. Specialized external services exist for each firm for financing through the stock market, international business consulting, and management training. The success of spinoffs, like Expedia, and startups, like RealNetworks, ushered in venture capital funding in the mid-2000s (Markoff, Market, and Bridge 2008). This secured access to capital for a small ecosystem of startup firms with minor working relationships with Amazon and/or Microsoft but that may include former employees who used the Amazon or Microsoft technologies in building their own software.

The Seattle region has grown, experiencing high rates of in-migration from within the US and internationally (Jurjevich and Schrock 2014). It has become a place for unskilled labor to become skilled and to enter the local labor market, creating another pipeline for Microsoft and Amazon.
I learned English and how to program at community college. It was easier to learn C... I transferred to Georgia Tech to get my degree... I didn’t really ever talk to anyone there, but I moved to Seattle to do Microsoft Certification and got a job at Microsoft. It’s what I always wanted to do.\textsuperscript{13}

At the same time, Seattle firms hire experienced international workers.

I worked for three years at a IT firm in India. We weren’t making software of our own, we were making software for our clients ... the quality of work wasn’t what I wanted to do. I had heard from friends and relatives about the kind of work you do here [at Amazon]. I decided to pursue my masters in the states... then I applied to Amazon.\textsuperscript{14}

Microsoft and Amazon dominate the regional labor market, and each firm has its own internal labor market. Some employees go from one firm to another, and four interviewees in Seattle had experiences working in other regions with large software agglomerations.

It’s common. A good one third of the people I work with [one of Seattle’s large firms] a good one-third have worked at [the other large firm]. A lot of workers at [large firm] went to Silicon Valley, a lot come back.

Workers are connected through exchange, and not necessarily through events that are standard practice in other regions like informal co-hacking events (often branded as meetups, sometimes user groups).

What’s a meetup? ... oh well, those kinds of events happen informally inside the different groups, plus we have the [customer facing developer event]. I

\textsuperscript{13}Seattle developer at a large firm. Interview, 11/19/2015, D29, transcript.

\textsuperscript{14}Seattle developer at a large firm. Interview, 10/27/2016, D27, transcript.
don’t go, but I encourage all my developers to go... I thought user groups were just for open source.\footnote{Seattle principal at a large firm. Interview, 11/20/2015, D30, transcript.}

Another product manager, when asked about meetups, explained that the utility of meetups was not for skill-building.

I’ve been to one but that one was organized by a friend, and I was kinda, sorta looking for another job. I think one of the reasons that Microsoft employees don’t go to meetups as much is because they mostly happen in the heart of Seattle, and Amazon is there. They fit the culture of Amazon better – Amazon churns through new college grads.\footnote{Seattle developer at a large firm. Interview, 10/01/2016, D24, transcript.}

And yet, another software developer, working remotely for a large Florida company, laments the lack of local relationships between software workers.

When I travel for work, I love going to meetups. There’s just not the setup to do it, and traffic, and we have to go from the eastside to Seattle or vice-versa.\footnote{Seattle developer at a large external firm. Interview, 10/04/2016, D25, transcript.}

Workers gain formal experience through certification or internal training. Microsoft has long had a strong connection with the UW and later connected with Bellevue Community College (now Bellevue State College) to develop its Microsoft software certification program to generate local skill. Microsoft also has an extensive internal training program. Amazon does not have a similar external training program but does provide time to workers to learn on their own and has internal software and business courses.\footnote{Seattle developer at a large firm. Interview, 10/27/2016, D27, transcript.} Both firms have been scrutinized for their labor practices. The region has strong labor orientation (Van Jaarsveld 2004). The oldest hub firms in the region are in
the forestry and fishing industries and have strong labor movement histories. Boeing, younger than its extractive industry counterparts, also has a strong history of labor organizing.

When Microsoft began hiring programmers on a temporary or agency basis, denying access to the employee stock purchase plan, employees sued and began an effectively unsuccessful unionization effort. Filed in 1992, however, the lawsuit wasn’t settled until 2000, when Microsoft agreed to pay $92 million in damages (Van Jaarsveld 2004). Microsoft instituted new rules around temporary workers but has since moved away from this practice.\(^\text{19}\) In 2016, The New York Times featured an article with over 30 interviews of Amazon office employees (as opposed to warehouse employees), documenting the high expectations and unforgiving workplace (Kantor and Streitfeld 2015). Yet only one of those interviews was with a software worker, one who left the company in 2006. The exchange fueled a series of posts between Amazon leadership and the newspaper’s editors. Amazon attempted to boost morale after the initial article by extending parental leave to six weeks.\(^\text{20}\)

Amazon and Microsoft software workers, as opposed to other office workers, were aware of the article, but agreed that the team they worked for was demanding but rewarding.

> I like it a lot. It’s demanding. It’s more intellectual.\(^\text{21}\)

> The company is exploding. There are a lot of growing pains ... there are really high expectations.\(^\text{22}\)

One employee compared the two cultures.

---

\(^{19}\) Seattle product manager at a large firm. Interview, 10/01/2016, D23, transcript.

\(^{20}\) Seattle developer at a large firm. Interview, 10/01/2016, D24, transcript.

\(^{21}\) Seattle developer at a large firm. Interview, 10/27/2016, D27, transcript.

\(^{22}\) Seattle developer at a large firm. Interview, 10/19/2016, D26, transcript.
They’re different vibes. Seattle is more ... mature. People have lives and more perspective. Yet the work is hard, but we also know that overwork isn’t ... it doesn’t translate to a better product. I don’t give my developers a lot of pressure, but I know they know my high expectations. It’s very different than how we used to operate.23

As a hub-and-spoke region, the technological practices fit with expectation. The firms value stability and compatibility with existing platforms and products.

For my team it’s about adopting modern engineering techniques and adopting best practices. We don’t care about the latest ruby framework.24

New software, new tools, and new software libraries are still important. Workers benefit by introducing new software and demonstrating its compatibility with the existing platform. This also provides job security for that individual worker.

The pressures ... you absolutely need to stay up to date ... you need to know the kinds of questions a potential candidate might ask you, and you need to know the language a candidate might solve a problem with... If you propose a better way of doing it, you’ll stand out. If you propose a new way, you should know it. And you’ll be in charge of it if it’s successful.25

Front-end web development changes at a faster rate than back-end operating system or cloud infrastructure. Developers working on the online portions of hub-and-spoke firms are still exposed to the technical churn.

Look man, I work in [***ing] front-end. My skills are outdated every three months. Today it’s in node.js, I learn it, then it’s Angular, I learn it,

---

23Seattle principal at a large firm. Interview, 11/20/2015, D30, transcript.
24Seattle product manager at a large firm. Interview, 10/01/2016, D23, transcript.
25Seattle developer at a large firm. Interview, 10/27/2016, D27, transcript.
then it’s some other stupid technology with another stupid name... then we
reconfigure the team to do ... front-end and back-end more and more.26

This quote also reveals that while they create software to automate tasks in other firms, managers also use software libraries to improve efficiency in their own workforce. Developers throughout my Seattle interviews echoed the expectation to use code to produce better code or to produce code more efficiently. In the previous chapter, a product manager even explained that the ideal programming candidate exhibits these behaviors.

The region is inextricably tied to Microsoft and Amazon and their engagements in emerging markets. The software agglomeration benefits from the continued support from UW, small spoke firms, and branch locations of firms like Adobe. The dominance of the two hub firms suggests a growth and innovation pattern based on capital-intensive acquisition. One developer worked at a company when it was acquired by a Seattle firm and chose to move to Seattle. Moving into new markets necessitated the adoption of new technologies, both emerging as important market makers in the fourth software period. At the same time, Microsoft and Amazon continually acquire new firms to protect their core business.

5.3.2 Portland, Oregon: a Marshallian software district

With its plethora of small- and medium-sized software firms and its well-connected software workers who work for both local and external firms, the Portland region represents a Marshallian industrial district structure. Researchers not located in Portland may skip over the region when reviewing software agglomerations, especially if evaluating the region on the 2009 or 2015 data in Table 5.1 and Table 5.2. Even in 1970 and 1980, the region may appear unremarkable in measures of software production activity. Portland and Austin have similar sized software agglomerations through the

26Seattle developer at a large firm. Interview, 10/01/2016, D24, transcript.
2000s, but Austin’s software agglomeration begins to grow faster relative to other regions in 2015.

Portland stands out for two other important reasons. First, among the 51 metros over with populations over a million in 1970, Portland had a small software agglomeration by count of software workers (36th) and a small agglomeration as a share of its workforce (44th). Since then, the software agglomeration in Portland, like Seattle, performed better relative to its peers in every period. As of 2015, Portland has the 22nd largest software agglomeration and the 14th highest share of its workforce in software production activities. Surprisingly, this rise comes without capital investment from government or local institutions of higher-education. Portland has a high-tech hardware agglomeration, but it has been historically separate from software production activity. Researchers embedded in Portland might also point to the second reason. In the 2000s, Portland gained a reputation among software workers for being open-source – the opposite of Seattle’s proprietary-software hub firm, Microsoft. Portland became of the home of Linus Torvalds, the creator and (at the time) principal developer of the Linux kernel, the core to all Linux-based operating systems. Torvalds was an important figure in this time period, as Linux became the operating system of choice for internet and cloud-based servers. Portland was also where Ward Cunningham, the inventor of the idea of the “wiki,” most famously used in Wikipedia, developed the idea. Alongside other events in the region, this orientation toward openness and interconnectedness helped Portland’s software agglomeration grow, albeit slowly, without the common ignition ingredients of other software agglomerations.

The roots of the Silicon Forest, however, go back even further to the 1930s and 1940s when a handful of electronics firms and a radio research facility located in the area. (Mayer 2012, 79)
Founded in Portland in 1946, Tektronix quickly became the world’s foremost supplier of oscilloscopes. It had a dominant market share for what at the time was a critical piece of technology for designing, manufacturing, and servicing, all manner of electronic devices, including computers. Tektronix was very profitable, and plowed much of its profit back into new product development and design, in the process attracting and retaining a cadre of talented engineers. (Cortright and Mayer 2000, 9)

In 1944, Douglas Strain founded Electro Scientific Industries (ESI, originally Brown Engineering) in Portland’s southeast. In 1953, Strain and three engineers invested a small amount of money into the firm. A focus on quality and frugality propelled the firm to create respected and inexpensive resistance-measuring instruments that were needed for manufacturing. Portland was unwilling to work with ESI to find suitable land with the city limits at the time. As a result, ESI moved to a science park in the suburban Washington County, where it hoped to attract other high-tech firms (Dodds and Wollner 2000).

Capturing the oscilloscope market, Tektronix had a similar upward trajectory. Founded as Tekrad in 1946, the firm produced oscilloscopes – a core component in manufacturing and in the design of new electrical instruments. In the 1950s, the firm followed ESI to Washington County, and by 1960, it had become Oregon’s largest manufacturing firm with more than 20,000 workers. Counter to traditional hub-and-spoke regions, both ESI and Tektronix had strong interconnections with one another and the founders had strong connections to other high-tech regions, including Silicon Valley and Boston (Dodds and Wollner 2000).

The history of Portland hardware firms, like Tektronix and ESI, sets the stage for Intel’s move to the region in the 1970s. Both firms dedicated significant firm resources
to the education if its employees, an important role in the region in the absence of a large, research-oriented university. Mayer (2005) calls Tektronix a “surrogate” university because it filled this role. Tektronix publicly stated its commitment to reinvesting in research and development and even offered classes to employees as a human resources policy. In the late 1950s and early 1960s, a time in which software was connected almost exclusively to military and institutions of higher-education (both of which were missing from Portland), Tektronix began Tek Labs and Tektronix Education Program (TEP). Tek Labs was the research-and-development arm of Tektronix that invested profits into research for new products. TEP was a week every year in which employees could take classes instead of working. In 1974, TEP’s student body was larger than enrollment at local universities, Reed College and Pacific University, and offered 99 classes.

Table 5.5: Industrial composition of software workers in the Portland region, ACS 2011–2015.

<table>
<thead>
<tr>
<th>Industry</th>
<th>n</th>
<th>% of software labor</th>
<th>% of labor force</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer systems design and related services</td>
<td>13,066</td>
<td>30.13%</td>
<td>0.56%</td>
</tr>
<tr>
<td>Electronic components and products, n.e.c.</td>
<td>4,146</td>
<td>9.56%</td>
<td>0.18%</td>
</tr>
<tr>
<td>Colleges and universities, including junior colleges</td>
<td>1,458</td>
<td>3.36%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Hospitals</td>
<td>1,312</td>
<td>3.03%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Management, scientific and technical consulting services</td>
<td>1,169</td>
<td>2.70%</td>
<td>0.05%</td>
</tr>
<tr>
<td>Insurance carriers and related activities</td>
<td>1,161</td>
<td>2.68%</td>
<td>0.05%</td>
</tr>
<tr>
<td>Computer and peripheral equipment</td>
<td>1,056</td>
<td>2.44%</td>
<td>0.05%</td>
</tr>
<tr>
<td>Banking and related activities</td>
<td>916</td>
<td>2.11%</td>
<td>0.04%</td>
</tr>
<tr>
<td>Total number of software workers in labor force</td>
<td>43,369</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total number of workers in labor force</td>
<td>1,367,515</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Both firms enjoyed growth through the 1970s. Portland grew its hardware high-tech sector again in 1974, when Intel located a memory manufacturing facility in Washington County, based on encouragement from Tektronix’s Board of Directors and relying on its relationships outside the region. Intel’s primary restriction was that it needed to be within a two-hour flight from its company headquarters in Silicon Valley. The Portland region was idea because of its local labor market’s talent and inexpensive workers. Intel’s facility manager was not concerned with the lack of a strong university presence.

“[W]e only wanted quality workers and power because it was a production
site” (Interview with Keith Thomson, Intel Facilities Manager, 2001 as cited in Mayer 2012, 100).

Intel’s decision to locate in Washington County spurred smaller supplier firms to co-locate in there in the late 1970s. By 1980, however, Intel had moved out of the memory-production market. According to a human resources study performed in the 1970s at Intel’s Silicon Valley location, potential workers were not attracted to working at the headquarters, and Intel was struggling to retain its engineers. Intel management sought secondary locations, and an engineering manager responsible for the iAPX 432 microprocessor leveraged the corporate policy in 1977 to move his team closer to his family. The microprocessor – Intel’s first 32-bit microprocessor – was a commercial failure, but it resulted in significant intellectual property. Later in the 1980s, Intel designed a second chip in the region – the RISC 80960 – which followed the same pattern, resulting in commercial failure but yielding significant intellectual property. Neither chip was based on the Intel 16-bit 8086, or even x86 compatible, which was necessary to run on the growing Intel and MS-DOS platform. Still, The Washington County team’s strong research and development attracted other Intel teams to the region, who were eventually able to apply insights from the previous two chips to develop the DX2-variant of the 80486, and later, the 80586, commonly referred to as the Pentium. The success of the 1993 Pentium and the 1995 Pentium Pro market successful Intel chip fully designed outside of Silicon Valley, further, Intel also manufactured the Pentium Pro in the region. The Pentium chip was a market success and brought prominence to Intel’s Portland-region location, further fueling the locational decisions of national and international suppliers (Dodds and Wollner 2000; Mayer 2012).

Intel’s growth in the region coincided with the decline of Tektronix and ESI. All three firms were responsible for spinoffs during the 1970s, 1980s, and 1990s (Mayer 2012).
MentorGraphics, founded in 1981, was one of the most successful spinoffs, producing design automation hardware and software. For some of the same reasons that Intel moved to Portland, MentorGraphics pitched in a job ad in the local newspaper: “Engineering and marketing opportunities are located in the clean air, uncrowded, affordable lifestyle of Portland, Oregon and other locations” (Rogoway 2016, emphasis own).

Figure 5.3: MentorGraphics job wanted ad in The Oregonian, 1982.
Two other startups from this era warrant mentioning. Central Point Software (CPS), which, founded in southern Oregon, moved to suburban Portland to take advantage of a larger labor market. Their flagship product, released in 1983, allowed users to duplicate floppy disks, even those with hardware copy protection. Intentional or not, it’s worth noting that the software circumvented some of the software licensing issues of the time by forcing the disk drive to ignore the copy protection. Of course their software was not free itself. They created other utilities in the Portland region before Symantec acquired and shutdown CPS in 1994. Two former CPS employees founded a new company, WebTrends, to take provide analytics for the websites of the fledgling online retailer business. WebTrends went public in 1999, then went private and sold to Houston-based NetIQ in January 2001 for $86 million. WebTrend founders publicly and repeatedly refused venture capital investment in their firm – instead seeking to go public without the influence of capital influx – much to the shock of venture capitalists (Woodward 2001). I highlight these two examples because they are indicative of the counter-culture ethos of many, not all, of Portland’s tech workers.

WebTrends was among a small set of firms in Portland whose timing allowed its founders to take advantage the IPO and later acquisition when the firm went private. A long-time Portland developer put it this way:

There were firms, but Portland is classically a B2B [business-to-business] town, we don’t do big flash everything tech ... The problem was, even though [some] firms went public, the market softened before people made wealth. Founders, a few founders made money, but we didn’t see that massive wealth creation like in Seattle... there’s no Jeff Bezos of Portland. There’s no Mark Cuban of Portland. That has slowed the growth of Portland because we didn’t have the capital to put back to work.27

27Portland veteran developer. Interview, 10/01/2016, D19, transcript.
Other developers in Portland at the time made similar comments. Portland’s software production activities were ramping up, and when the dotcom bust happened, the speculative investment in software firms evaporated, firms could not continue to employ workers. Portland had not already had the multiple rounds of investment into firms, wealth generation, then reinvestment into new firms that other regions had enjoyed during the dotcom bubble. This also prevented the region from learning and skill-building in the software startup scene.

Back then all the activity took place in Hillsboro. I bounced around a bunch of those companies. The [software] scene back then, there was activity, but it was hidden, there were very few companies that stood out as tech startups. Then the dotcom thing happened, and startup became part of popular culture and Portland didn’t see any upside in that. It wasn’t until the dotcom crash that open source really seemed to gain footing and prestige around Portland... there’s this culture of craft, making it as best possible ... and this culture of collaboration and willingness to help one another.²⁸

While the original high-tech hardware hub-firms in Portland grew, then began to decline, Intel’s branch-plant location grew to be its largest facility and acted more like a headquartered hub-firm, producing more research and development than Intel’s other locations combined. Software production activity in Portland was embedded in other industries and an absence of large successful firms that produced local capital meant that Portland didn’t have the thriving software agglomeration that other regions had. Portland’s exposure to software as commodity came from the proliferation of the Intel chips that ran the software, not the software itself.

²⁸Portland veteran developer. Interview, 10/01/2016, D19, transcript.
The 2000s mark a change. Workers in Portland had technical skills, but lacked the know-how to attract venture capital and thus rapidly expand. Instead, in the time after the dotcom bust, new startups in Portland relied on open source technologies to build technical solutions, often web oriented. Portland’s low-level hardware orientation meant that Perl was a common language used across industries. In 1999, Portland software workers formed a local user group, Perl Mongers, to exchange Perl tips and tricks and contribute to the Perl project.

And 30-40 people would show [to Perl Mongers in 1999-2000]. And that was the cutting edge. I was writing stuff in Perl at Intel and people thought that was weird. I was working - I was in network engineering and I was automatically configuring routers and switches and stuff like that. My co-workers on the IT side were still manually loading things from Windows laptops. I was on the edge of change at Intel. 29

Important Perl contributors and developers also happened to live in the region, some working for external firms. One had created a Perl module that automatically configured routers that gained popularity among those doing networking. It became industry standard. The regions workers, without software oriented firms to steer or direct software development energy, found their own outlets for engineering through the internet and ultimately toward open source projects. Ward Cunningham was among a list of popular local developers, as well as contributors to the Eclipse, parts of different Linux distributions, and open source packages for software development. Intel was simultaneously working seeking to expand its footprint in server processors and in networking hardware. Linux was the de facto operating system of the internet and thus

29 Former Intel developer now at a large external firm. Interview, 10/08/2015, D15, transcript.
Intel began a small group of software workers that contributed to the Linux kernel, ensuring compatibility with Intel hardware.

Open source was a thing. I think it was. I think it was a distinguishing mark for the tech community, it brought people together for a long time.\textsuperscript{30}

In 2001, just one year after only one year in SF, the Open Source Development Labs (OSDL) moved its headquarters to Beaverton. The (OSDL) was a large consortium of firms invested in the continued development of Linux and the Linux kernel and included Computer Associates, Fujitsu, Hitachi, Hewlett-Packard, IBM, Intel, Nippon Electric Corporation. By 2003, after two poorly attended conferences in San Diego, O’Reilly Open Source Conference (OSCON) moved to Portland in hopes the Perl community would bolster its numbers – and it did. In fact, OSDL’s move prompted Torvalds to move to Portland, as mentioned earlier. Portland became a hotspot of development, and firms sought to use open source to develop products.

The Portland open source community was well organized and well connected via community email lists. Starting in 2004, an influx of firms moved to Portland or moved their engineering workers to take advantage of the open source milieu. This list includes Jive, New Relic’s engineering headquarters (which hired Ward Cunningham), IBM’s Linux development group, Janrain, Oregon State University’s Open Source Lab, Puppet Labs (now Puppet), and Urban Airship (Rogoway 2011). Local firms worked with local government to form a notable incubator project. Portland’s small firms grew to medium firms. More recently, the community has sought to stay connected and pursue issues that collectively affect software workers in the region by writing open source calendar tracking for the region (somewhat reject the branded meetup), forming conferences like OSBridge

\textsuperscript{30}Portland developer at an large external firm. Interview, 10/08/2015, D15, transcript.
and PyDX with a focus on equity and diversity training, and events like WriteTheDocs aimed at improving documentation in open source software to reduce barriers to entry.

Portland’s accidental collection of Perl developers made Portland remains a “sticky” place for open source, yet the region continues to be in a healthy turmoil. In 2007, the OSDL merged with the Linux Foundation and moved to SF. In 2016, OSCON permanently moved from Portland to Austin. Portland firms and software workers may have missed opportunities in the software as commodity growth period, but leverage the absence of capital to build strong relationships within the community. Building off its experience in networking and its business to business history, this has the right business to business knowledge and as new firms grow. A private equity firm acquired Jive in 2014. New Relic, still headquartered in Silicon Valley continues to expand. Bank Simple, a bank and financial software company, moved its headquarters to Portland. Puppet continue to grow. Other than Intel, the original group of hardware firms in the region continue to retract (Rogoway 2014).

The region certainly resembles a Marshallian industrial district. The growth and structure of Portland is dependent on the network of small and medium sized firms, aided by Intel’s continued and significant branch-plant adjacency, to continue to innovate. The structure is still dominated by small and medium firms, and the labor market is internal to the district. The region continues to grow as it attracts new migrants (Jurjevich and Schrock 2012; Jurjevich and Schrock 2014). According to my interviewees, Intel, while enormous, does not dominate the software labor market, but instead relies on the district as well. Washington County has long been supportive of large firm investment from companies like Intel, Nike, and Tektronix. Portland, initially unwilling to work with Tektronix and ESI in the 1950s, now cooperates with technology – and software – firms, incorporating them into its economic development plan. The Portland Development Commission even donated seed money to the local incubator and the
OSBridge conference.

Workers have strong intra-district relationships, and because of the internet connected nature of open source, have formed organizations and conferences to build and protect its workers. Two successful annual conferences were already mentioned. A new community initiative, SafetyFirstPDX, aims to develop code of conduct language so that any conference organizer can have access to a variety of language and guidelines around conference organization. These strong local connection aim to reducing risk to the district, not necessarily to the firm, and provide specialized skills, business and technical training, and even advice on how to navigate, or reject, venture capital investment.

Early third generation languages like Perl may have been a reason to come together, but the region, and the language, are not as prominent as it once was. Perl provided important flexibility in hardware infrastructure, but the region has strong Python (PyDX) and Ruby orientations. The region uses these tools to development tools for other businesses, relying on its strong worker relationships, the interstitial space between existing firms, for sources of innovation. Two interviewers explained that Portland is a place of “accidental innovation” because software workers in Portland are trying to solve their own software development issues. Nike, and the US headquarters of Adidas, are firms in the region that employ small but influential groups of software developers. Wearable technologies, a small, but emerging market, employs software workers in both firms, overlapping with hardware technologies researched and produced locally at Intel. Several small startups have emerged in the wearable technologies space, some even claiming that their startup idea came from the founder’s own needs for fitness tracking. Interviewees shared a strong narrative between accidental software innovations and adjacent industries.\textsuperscript{31}\textsuperscript{32}

\textsuperscript{31}Portland developer at an large external firm. Interview, 10/08/2015, D15, transcript.\textsuperscript{32}Portland economic development practitioner. Interview, 10/02/2015, D14, transcript.\textsuperscript{33}Portland veteran developer. Interview, January 16, 2016, D19, transcript.
5.3.3 Austin, Texas: a region in flux

The Austin region does not match a single type from Markusen’s industrial district typology. At first blush, Austin appears to be solely a state-sponsored region. The Texas State Capital building looms large over downtown and to the north is the sprawling, but still urban, University of Texas campus. For these reasons alone one might miss the other software production activities in the region. I will argue that Austin is a mixture of state-sponsored and branch-plant types and that this mixture is of on-going change that may develop into a Marshallian industrial district mixture. At the time of writing, Austin’s industrial type is in flux and there is significant variability and uncertainty in future pathways of development. Even if Austin’s branch-plant and Marshallian components evaporate, Austin has the advantage of obdurate state-sponsorship.

Austin has positioned itself as a technopolis since at least 1957, when the Austin Area Economic Development Foundation, a public-private partnership, aimed to leverage its state-sponsored assets and economic structure to attract new high-tech manufacturing.

“A team of public and private leaders built upon UT’s [University of Texas] research programs by attracting manufacturers of electrical and scientific equipment. Austin’s success in the 1960s and 1970s in making the transition from government to high-technology manufacturing was built on the traditional factors of relatively low-cost land, a high-skilled workforce at reasonable wages, and a desirable quality of life” (Humphrey 2013, 17 cited in Straubhaar et al. 2012, 67).

This process put into motion a series of high-tech manufacturing satellite-branch plant openings that included IBM typewriters and Texas Instruments chips in the 1960s and then Motorola chips in the 1970s. These branch plants and the expansion of UT’s computing and engineering, while not exclusively software-oriented, are certainly complementary...
to the rise in software production activity in Austin. In the early 1970s, the US Navy contracted UT’s expanded engineering research team to produce better components to collect and analyze data from naval instrumentation. In 1976, research faculty led by James Truchard founded National Instruments to create better equipment without the constraints of working within UT or being a government contractor. Growth for National Instruments was slow during the 1970s and 1980s, but this marked one of the first successful spinoffs from UT’s investment in engineering and computing research. In 1979, Silicon Valley-based Advanced Micro Devices (AMD), and Intel’s primary competitor, also opened a semiconductor branch-plant facility in the region (McComb 2010; Humphrey 2013).

In software’s 1970 and 1980 business-based period, Austin supplied the semiconductors that powered business computing. The amount of software production activity, compared to other production activity and relative to other metros, catapulted Austin into the top 7 regions by software labor force share starting in the 1980s. Further, the administrative tasks of both the state capitol and the home of the University of Texas system both required mainframe computers to process the everything from taxes, to admissions, and payroll for the growing state. This further added growth of software production activity in the region. Yet, the linkages between software production activity related to branch-plant chip manufacturing remained separate from software production activity in the state capitol which remained separate from the software production activity at the University of Texas.\footnote{Austin product manager in government. Interview, 10/30/2015, D7, transcript.} Weak links between these silos fit with expectation of both branch-plant and state-sponsored district types.

Austin’s computer hardware sector continued to grow in the 1980s and 1990s into the software as commodity period. In 1982 amid increasing competition from Japan, the Microelectronics and Computer Technology Corporation (MCC), a consortium
of declining chip manufacturers that sought to save costs by joining efforts in research for chip design and manufacturing, chose Austin to be its home because of its strong record of public-private partnerships. Austin beat Silicon valley as a potential location and San Antonio endorsed Austin’s bid. Local businesses, combined with UT and the City of Austin, provided incentives of $62 million. Later, in the 1988, Sematech, another research consortium of semiconductor manufacturers originally sponsored by the Department of Defense to engage in research to design chips for the US government. In 1983, the Dean of UT’s College of Business founded a think-tank called the Innovation, Creativity, and Capital Institute (IC²). The IC² promoted the high-tech, albeit non-software focused, economic success of Austin as a model for other regions based on the economic development relationships local government, universities, and the private sector (Straubhaar et al. 2012; Humphrey 2013). This top-down model values the economic development relationship rather than the relationships between workers and conveniently ignores the important role that the state capitol played in contributing in the demand for high-tech workers.

In 1984, former UT student Michael Dell founded Dell Computer Corporation outside Austin in suburban Round Rock. The company grew steadily through the 1980s, but didn’t grow rapidly until Dell started selling computers directly to customers in 1996 through its website which allowed visitors to customize their orders. Dell then surpassed Compaq as number one computer retailer in 1999, and then again became number one retailer when Hewlett-Packard purchased Compaq in 2000. Dell grew fastest in the early 2000s, coinciding with the release of Windows XP and their expansion into low-end or small office servers. National Instruments released LabVIEW in 1986, a graphical programming language, which allowed engineers to program advanced data acquisition instrumentation by creating software flows that looked mimicked the flows of a circuit board. National Instruments catered instruments for military research, academic research,
Table 5.6: Industrial composition of software workers in the Austin region, ACS 2011–2015.

<table>
<thead>
<tr>
<th>Industry</th>
<th>n</th>
<th>% of software labor</th>
<th>% of labor force</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer systems design and related services</td>
<td>22,127</td>
<td>40.27%</td>
<td>1.17%</td>
</tr>
<tr>
<td>Computer and peripheral equipment</td>
<td>3,810</td>
<td>6.93%</td>
<td>0.20%</td>
</tr>
<tr>
<td>Electronic components and products, n.e.c.</td>
<td>2,381</td>
<td>4.33%</td>
<td>0.13%</td>
</tr>
<tr>
<td>Colleges and universities, including junior colleges</td>
<td>1,972</td>
<td>3.59%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Electronics stores</td>
<td>1,956</td>
<td>3.56%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Management, scientific and technical consulting services</td>
<td>1,529</td>
<td>2.78%</td>
<td>0.08%</td>
</tr>
<tr>
<td>Software publishing</td>
<td>933</td>
<td>1.70%</td>
<td>0.05%</td>
</tr>
<tr>
<td>Non-depository credit and related activities</td>
<td>909</td>
<td>1.65%</td>
<td>0.05%</td>
</tr>
<tr>
<td><strong>Total number of software workers in labor force</strong></td>
<td>54,943</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total number of workers in labor force</strong></td>
<td>1,055,142</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


and manufacturing, effectively connecting its fate to Austin’s high-tech economy.

Table 5.1 and Table 5.2 show that Austin failed to maintain its comparative growth between 1980 and 1990, but maintained a significant software agglomeration. Austin again grew its relative share between 1990 and 2000, but the 2000s saw the decline of much of Austin’s semiconductor manufacturing as Sematech relocated to Albany, New York, MCC shutdown, and both AMD and Motorola closed their Austin production facilities after failing to compete with Intel and suffering hard from the retreat of computer purchases related to the dotcom bust. Dell continued to grow because of its expansion into new markets. Austin, as a region, continued to grow and its reputation as a technopolis along with it thanks to the IC² principles of economic development relationships between local government, local university, and the private sector. In terms of software production
activity, this also marks the period when the complementary branch-plants semiconductor district began to falter and replaced with the computing-based hub-and-spoke district of Dell. Dell and UT dominate the hardware labor markets, but only UT dominated the software labor market (McComb 2010).

Importantly, in the late 2000s, the same model that attracted branch-plant hardware production began attracting branch-plant software production. Google, Twitter, Dropbox, Microsoft, Apple, have locations in Austin. It’s not uncommon for large firms like these to open locations outside their headquarter region, but these locations often don’t contribute to the core production, software or otherwise, activities of these firms. In some instances,
Microsoft and Google have locations that are co-working spaces for workers across teams that are not located in the firms headquarter location. Apple in Austin is an exception in this case, much like Google and Adobe are in Seattle. Apple seeks to tap the local research and design labor market around semiconductor chips and to develop its own chips for its computing devices. More in line with expectation Apple, along with Google, have customer and technical support facilities in Austin.

While these small branch-plant facilities opened in the 2000s, the 2010s mark the potential emergence of connected workers in small firms interacting with software production branches. In the 2010s, several small Austin startups began to grow and become the cornerstones of the region’s startup scene. These includes BazaarVoice, WPEngine, and RetailMeNot. Software production activity in these firms was still minimal compared to the larger region, but were explicitly software, not hardware. This period also marked the emergence of high-profile branch-plant opening from Australian firm Atlassian. Atlassian makes productivity software for software developers and moved its entire software production to Austin. Further, Dell’s continued expansion into cloud computing spurs software production specific to the operations data centers.

It’s odd because ... it’s hard to say, because on one hand people are like the tech scene in Austin is so big and so great. On the other hand, yes, but it’s never going to be like Silicon Valley, it’s like a shave of San Francisco, like you could never compare it to those cities, but compared to a lot of cities there’s some good startup initiatives. I wouldn’t, I guess I could say there’s some enterprise kinda things going on.... it’s a small tight knit bunch with a lot of new people. ... knowing people from other cities, I get the impression that Austin is really supportive.35

35Austin developer at a large external firm. Interview, 11/05/2015, D1, transcript.
The influx of software workers at software firms may have trouble connecting to the software workers in government, at UT, and in hardware-based hub firms. One manager put it bluntly, confirming what other public sector software developers had already said:

I'll say that the actions you [worker] have with the broader community is up to them. It’s not particularly encouraged or discouraged. Now, obviously, you can improve your skills by doing that and [subordinate developer] took that tact. I do very little of it. I do different things when I leave. ... Sometimes with government agencies ... there’s a gulf between what’s going on in government and the private sector. Where it’s going now, we might be more mainstream. It’s almost surprising how much little of [interaction with programming community] there is.\textsuperscript{36}

One entrepreneur and developer organizes a monthly meetup that ranges from 40 to 75 people per month in an advanced third generation language. This developer advocates for Austin and the strong labor demand.

There are 110 people moving to Austin everyday. Austin is a place people want to be and [firms are] struggling to fill roles. Get them skills. Get them jobs. It’s social and technical. It's friendly, we share, it’s good for your team and good for your career and people find jobs and co-founders.\textsuperscript{37}

Austin doesn’t have the built environment and clustering of downtown software workers, but this meetup organizer explained that there’s a lot of parking downtown for people coming into town. Tuesday’s after 6:00 p.m. parking is free and I’m told that this warrants a lot of meetups to happen on Tuesday nights. Another developer mentions that Austin is a growth hub. Firms have to use code sharing technologies, new languages,\textsuperscript{36,37}

\textsuperscript{36}Austin developer on programming in the 1980s. Interview, 10/30/2015, D7, transcript.
\textsuperscript{37}Austin developer and entrepreneur. Interview, 11/02/2015, D11, transcript.
and prototyping-based production methods. Amazon’s cloud infrastructure is a standard among startups works well with the flexible languages that workers are using, allowing them to scale as necessary.

We have to stay up on new frameworks and new languages. Python and Ruby are vital to the success of our startups... we’re not going have a startup with a product in C++ [laughter] ... they don’t even teach that.\(^\text{38}\)

Above, the public sector manager mentions that they’re becoming more mainstream alluding to the recent switch from legacy programming languages to Python. This has led the subordinate to attend local meetups to gain experience. Certainly a mixture of technologies are present from legacy mainframe technologies to bleeding edge software languages. None of the interviewees outside the public sector were aware of the 500 software developers within UTs administration, 300 software developers within the state capitol, or the thousands of developer spread across teams in both.\(^\text{39}\) While these are small numbers in the near 55,000 software workers in Austin, they are larger than the well-known Atlassian development site and all are in the urban core.

Meetups are less about deep learning and more about interacting. [Everyone has the opportunity to participate.] Austin isn’t wrapped up [with] what company you’re at. We’re more open to sharing experiences.\(^\text{40}\)

Each of my interviewees explained that meetups is a place where skill-building can happen, but everyone benefits socially and that they are good places for mentoring or career-building. Common to the public sector interviewees, workers learn skills within the organization as opposed to having experience before starting. In fact, one public

\(^{38}\) Austin developer and entrepreneur. Interview, 10/27/2015, D4, transcript.

\(^{39}\) Numbers based on interview with high-level government-based project manager with experience in both location. Interview, 10/30/2015, D7, transcript.

\(^{40}\) Austin developer at an SME. Interview, 10/31/2015, D10, transcript.
sector organization uses the IBM Information Processing Aptitude Test (IPAT), licensed from IBM in the 1980s, to select candidates. Candidates are not required to have any programming experience. If candidates score high on the IPAT and perform well in interviews, they spend the first 3-6 months learning both a legacy language and Python.

The City of Austin has long supported investment in high-tech, but has only recently supported software production activity. Capital Factory is one of the largest tech incubators alongside Google for Entrepreneurs, both of which provide product advice, startup space, and meeting rooms conditioned upon acceptance into the program. Capital Factory also runs a selective and intensive 3-month coding school.

Not everyone has access to the software labor market or these coding schools. Austin has long had discussions about equity and technology, especially with the role of IC² has played in the region. Numerous local organization, partnered with international organization, seek to protect and develop the local software scene. Like in Portland, strong organizations like Women Who Code and local initiatives like Latina Hackers seek to provide the training to navigate the becoming, and advancing, as a software developer. However, the women interviewees lamented that these initiatives are often short lived, sometimes only having one meeting, before forgotten. Within the regions hardware firms and within the public sector, there exist women-only networking and support groups, they aren’t as specific women in software, and don’t provide networking opportunities to other software firms in the region.

Part of the purpose of these groups is to discuss working conditions and the types of expectations that software workers can or should communicate to their managers. Significant challenges still exist for Austin to develop a successful Marshallian software district.
I got tired of being told ‘your job is to execute’ everywhere I worked [in Austin]: we know how develop, but not many know how to manage.\footnote{Austin developer at an SME. Interview, 11/02/2015, D12, transcript.}

Branch-plant and hub-and-spoke hardware manufacturing firms employ large numbers of software workers with little connection to the rest of the city. Software-focused public sector production activity has weak, but strengthening, connections to software workers in small firms in the region. The small firms are well-connected through meetups and connections from gained at startup spaces. Small firms are dependent on new technologies and being ahead of competition in terms of technologies. The public sector organizations depend on stable, legacy software. There are exceptions to both.

Austin’s software production activity is strong, but its type is a mixture. Austin is certainly state sponsored, but there appears opportunity for a Marshallian district to emerge, and whether that Marshallian district interacts with existing software production, whether in the public sector or the workers embedded in industry throughout the region, remains to be seen. The successful agglomerations in Seattle, Portland, and Austin all have significant components of cloud computing. In Austin, Dell’s server business employs Austin’s largest visible group of cloud computing devoted developers in the region.\footnote{Austin developer and entrepreneur. Interview, 11/02/2015, D11, transcript.}

While doing field research in Austin at the end of 2015, two important events happened related to Austin’s software-adjacent production activity. James Truchard, CEO of National Instruments, announced his retirement at the end of 2016. National Instruments subsequently offshored all its local manufacturing, after offshoring a large portion of it in the previous year, and continues to retract its Austin workforce (Graar 2014; Andersen 2017). Second, Dell purchased EMC\footnote{EMC is not an acronym} and its Silicon Valley-based subsidiary VMware. Boston area-based EMC (now Dell EMC) surpasses global firms like IBM, Hewlett Packard Enterprise, and Hitachi Data Systems, to capture the largest share of...
the market in data-storage systems. The acquisition fits in line with Dell’s continued investment in servers and cloud computing as data-storage is a key component of data center operations. VMWare provides virtualization software, allowing a single computer to run multiple similar or different operating systems, a key component of large data center operations (Rockwell 2016). At the time of writing in 2017, it does not appear that Dell will relocate the software production activity components of these firms to Austin, or how this purchase will impact Dell’s Austin-based software production activity in cloud computing.

Austin is a region in flux – or perhaps, as Markusen, DiGiovanna, and Y. S. Lee says, it’s a “sticky mix.”

5.4 From Industrial Districts to Regional Outcomes

5.4.1 Comparing the observed industrial districts

Summarizing the work in the previous section, I highlight the different variations in industrial districts characteristics in Table 5.7. These characteristics form the basis of each region’s specific mode of software production. As firms and workers in each of these second-tier regions develop new software, they diverge across production (technologies, skill-sets, labor markets), social relations (class-based firm-worker relationships), and/or spatial organizations (industrial district formations). In this section, I analyze how the different modes of production identified in the previous section, and their historical trajectories, produce different sets of software occupations and income distributions in each region.

Again, software production’s low-materiality affords rapid cycles of development, feeding back into changes in the production of software and fueling regional divergence. The particular technologies impact what software is possible, firms organize the method
Table 5.7: Comparing industrial districts of Austin, Portland, and Seattle.

<table>
<thead>
<tr>
<th></th>
<th>Seattle</th>
<th>Portland</th>
<th>Austin</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Hub-and-spoke with strong university links, historical benefits from Boeing’s presence</td>
<td>Marshallian district with adjacent industries, benefits from Intel’s networking and IoT focus</td>
<td>Growing Marshallian district, historically public-sector district adjacent semi-conductor industry, benefits from Dell’s server software focus</td>
</tr>
<tr>
<td><strong>Government</strong></td>
<td>Caters to hub firms, strong connections to University of Washington</td>
<td>Recent acknowledgment of software (separate from hardware) and pursuit of economic development strategy</td>
<td>Recent acknowledgment of software (separate from hardware) and pursuit of economic development strategy, strong public-private partnerships</td>
</tr>
<tr>
<td><strong>Firm links</strong></td>
<td>Strong relationship between hub-firms and external firms, weak local relationships</td>
<td>Strong relationship between firms</td>
<td>Growing relationship between firms, and between public-sector and firms</td>
</tr>
<tr>
<td><strong>Labor links</strong></td>
<td>Weak relationships between workers outside of firm</td>
<td>Strong relationships between workers, strong regional worker identity</td>
<td>Growing relationships between workers, slowly accessing public-sector workers</td>
</tr>
<tr>
<td><strong>Labor market</strong></td>
<td>Strong relationship to Silicon Valley, dominated by Microsoft and Amazon, each having their own internal labor market</td>
<td>Strong relationship to both Seattle and Silicon Valley, workers connect with each other</td>
<td>Multiple: public-sector and private-sector, with increasing relationships, growing rapidly</td>
</tr>
<tr>
<td><strong>Intermediaries</strong></td>
<td>Few, strong internal firm dynamics</td>
<td>Strong, based on local events, local serving conferences</td>
<td>Strong, connected to Capital Factory, and local events</td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td>Historically proprietary, reliance on 3G+ languages, developing new libraries/languages for cloud infrastructure, connected to Microsoft platforms and Linux platforms</td>
<td>Historically open source, reliance on flexibility of newer 3G+ languages, strong ties to Linux</td>
<td>Reliant on newest open-source for competitive advantage, orientation toward 3G+, significant expertise of legacy languages in public sector</td>
</tr>
<tr>
<td><strong>Innovation</strong></td>
<td>Based on acquisition of small local or external firms, small research units internal to anchor firms</td>
<td>&quot;Accidental innovation&quot; related to software workers interacting across industry lines, in-between spaces/events important</td>
<td>Sponsored through public-private partnership of Capital Factory, new connections to UT and Dell</td>
</tr>
<tr>
<td><strong>Automation</strong></td>
<td>Make automation tools for other firms, use of automation tools on own workers</td>
<td>Makes automation tools for other firms and workers, adoption of automation tools in development by workers</td>
<td>Use of automation tools by other firms</td>
</tr>
<tr>
<td><strong>Fourth period</strong></td>
<td>Reliance on Microsoft and Amazon cloud performance, experimental languages</td>
<td>Small firms provide cloud infrastructure tools and developer tools, serving other software businesses</td>
<td>Growing startup scene leveraging cloud infrastructure, Dell to play increasing role, fall-back on public-sector</td>
</tr>
</tbody>
</table>
and direction of production, and workers provide the knowledge. Though, as we saw in Portland, workers have high levels of interaction outside of the firm. Software workers in Portland have higher loyalty to the district than Seattle’s workers, who tend to have higher loyalty to the firm. Austin’s industrial district is evolving from a public-sector industrial district, benefiting from the hardware industries in the region, into a Marshallian district. Its Marshallian district, like Portland’s, is relying heavily on open-source technologies and languages to benefit from the network externalities of open-source language adoption. The connection to open source further encourages networking and exchange between software workers in Austin, who form meetings, user groups, and local-serving conferences. This differs from Seattle, where events are either held within the firm or serve connections to other large firms. The Microsoft Build Developer Conference and Amazon’s re:Invent are both held in Seattle and are both aimed at drawing in other software firms and workers. Microsoft and Amazon have strong linkages to large firms outside the region but relatively weak relationships with spoke-firms in the region. The connections among spoke-firms are also weak. Made through their employees and their products, connections among small firms in Portland and Austin are stronger. Software workers at small firms in Austin are making connections to two large public institutions in the region.

While each region has strong connections to Silicon Valley – through industry linkages in Seattle, labor linkages in Portland, and branch operations and growing labor linkages in Austin – the labor market processes within each region are regionally distinct. Software workers, as part of an educated labor force, are among the most mobile workers, and my interviewees were no exception. While some had only worked in one of the three case regions, most had worked in multiple cities, and some even had experience working in all three locations and Silicon Valley. As expected, Seattle has internal labor markets, internal to each firm, but the two hub-firms both compete locally for top talent and draw workers from around the world. The labor market in Portland and Austin is internal to the district.
and based on making connections among existing software workers. Community groups and user groups in these regions organize technology-specific meetups and events, which serve as labor market intermediaries for new workers. While the events provide space for accidental innovations, mentoring and other interactions at the events allow workers to build tacit skills, practice presentations, and get advice on resumes (especially tips on key words and phrases used to filter applications). Austin uniquely has a second labor market, internal to its large institutions.

Austin software production differs from Seattle and Portland in the generation of software languages commonly used. In Austin’s internal public-sector institutions, an established presence of legacy software requires special training programs on legacy languages, like NATURAL and the locally designed WebAgent, distinguishing the expertise in the region. Public-sector software workers are attempting to modernize to frameworks, like Django, and languages, like Python, but a proposal for a new set of servers and various off-the-shelf software packages—some developed in Seattle—threatens to bring automation and replace workers in public-sector institutions. As I showed in the previous section, other forms of automation in Austin attempt to regulate software worker tasks and time management through machine-learning.

Seattle software production differs from that of Portland and Austin in terms of open source and proprietary software development and use. The proprietary nature of software programming in Seattle forces the firm to seek innovation internally or through acquisition. Conversely, the networked nature of open-source software development requires interaction both locally and globally. Workers gain skills and expertise by exchanging knowledge locally. For workers and startups without capital investment, open source provides a way to enter the market without upfront technology or licensing costs. In Seattle, Microsoft and Amazon can invest in or acquire local firms and provide those firms with access to their proprietary platforms and technologies in pursuit of innovation.
5.4.2 Distributions of software worker incomes

These software production differences produce regionally specific labor market processes, confirming the idea that wage-employment relationships take on local characteristics (Peck 1996, 110). We then expect to see related differences in occupational structures and distributions of income. I show the income distributions of software workers in Figure 5.8 and Table 5.8. Seattle has a higher median income for software workers. The regional median income and the ratio of software median income to regional median income are included in the table, allowing comparison of regional variations in income and cost.
of living. Seattle software workers make more than their counterparts in Austin and Portland and make more relative to the regional median income. The even distribution of software worker incomes is in part due to the two hub-firms’ controlling incomes for large shares of software workers, using corporate policy as guidelines in promotion and salary increases.

In Seattle, more workers are in the middle of the income distribution. Portland and Austin both have larger shares of workers at the bottom of the income distribution curve. The middle of Austin’s income curve splits into two, or possibly three, income groupings, likely representing different groups of incomes set by a more rigid public-sector institution. In Portland, the large number of small firms gives the region’s distribution of income more variation and results in more workers in a larger middle. For each region, the 25th and 75th percentiles of income are approximately $30,000 above or below its median income. This suggests a common spread of incomes across all software occupations in all regions, even though variances exist between the regions in the middle, as noted earlier.

Certainly, regional modes of production affect incomes within regions. However, the income distributions across all workers can explain only some of the variation. For

<table>
<thead>
<tr>
<th>Table 5.8: Median incomes across Austin, Portland, and Seattle, ACS 2011–2015.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Median incomes of software workers, Austin, Portland, and Seattle, ACS 2011–2015.</strong></td>
</tr>
<tr>
<td>Median income, software workers</td>
</tr>
<tr>
<td>Median income, all workers</td>
</tr>
<tr>
<td>Software workers’ median income : regional median income</td>
</tr>
<tr>
<td>25th percentile income, software workers</td>
</tr>
<tr>
<td>75th percentile income, software workers</td>
</tr>
</tbody>
</table>

Author’s calculations from data gathered from IPUMS-USA (Ruggles, Genadek, et al. 2015) ACS 2011–2015. Software occupations include OCC codes 110, 1005, 1006, 1007, 1010, 1020, 1030, 1050, 1060, 1105, 1106, 1107, 1200, 1220. Data is based on income from wages (INCWAGE) and excludes income reported on 1099s.
Table 5.9: Software occupation classification, ACS 2011–2015.

<table>
<thead>
<tr>
<th>Code</th>
<th>Title</th>
<th>Classification</th>
<th>Median Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>Computer &amp; information systems managers</td>
<td>Managers</td>
<td>95,000</td>
</tr>
<tr>
<td>1020</td>
<td>Software developers</td>
<td>Engineers</td>
<td>90,000</td>
</tr>
<tr>
<td>1106</td>
<td>Network architect</td>
<td>Engineers</td>
<td>90,000</td>
</tr>
<tr>
<td>1200</td>
<td>Actuaries</td>
<td>Specialized Analysts</td>
<td>96,005</td>
</tr>
<tr>
<td>1007</td>
<td>Information security analyst</td>
<td>Specialized Analysts</td>
<td>80,092</td>
</tr>
<tr>
<td>1005</td>
<td>Computer research scientist</td>
<td>Programmers</td>
<td>74,085</td>
</tr>
<tr>
<td>1010</td>
<td>Computer programmers</td>
<td>Programmers</td>
<td>73,771</td>
</tr>
<tr>
<td>1006</td>
<td>Computer scientist &amp; systems analyst</td>
<td>Analysts</td>
<td>71,000</td>
</tr>
<tr>
<td>1220</td>
<td>Operations research analyst</td>
<td>Analysts</td>
<td>71,320</td>
</tr>
<tr>
<td>1060</td>
<td>Database administrators</td>
<td>Administrators</td>
<td>71,227</td>
</tr>
<tr>
<td>1105</td>
<td>Network &amp; computer systems administrators</td>
<td>Administrators</td>
<td>64,074</td>
</tr>
<tr>
<td>1030</td>
<td>Web developers</td>
<td>Coders</td>
<td>42,155</td>
</tr>
<tr>
<td>1050</td>
<td>Computer support specialist</td>
<td>Coders</td>
<td>46,454</td>
</tr>
<tr>
<td>1107</td>
<td>All other computer occupations</td>
<td>Other</td>
<td>60,000</td>
</tr>
</tbody>
</table>

Note: See Table B.1 in Appendix B for descriptions, incomes, and more information.


Further analysis, I take the fourteen software occupations from the ACS 2011–2015 and group them into eight classifications based on typical tasks and functions: managers, engineers, specialized analysts, programmers, analysts, administrators, coders, and all other occupations. Each classification, except managers and all other occupations, is the combination of two occupations. In Table 5.9, I list each occupation and its national median salary.

To gain further insight, I compare the occupational structure of each region by looking at the prevalence of each occupation classification. Table 5.6 and Table B.1 show these results. Two observations are obvious. First, the regional share of engineers for all regions, comprised of network architects and software developers, makes up more than half of
the regional software labor force. Second, Portland and Austin both have larger shares of coders, programmers, administrators, and analysts, all at the expense of engineers.

Microsoft had formerly split its software occupations into testing, development, and development in testing.

We went through a restructuring, and we found that the metrics that we had for measuring success at each position could be rigged ... you could have excellent metrics, but that didn’t translate into a better product. We wanted to make a better product.44

My interview subject continued to explain the metrics system combined with the rank ordering of employees on teams promoted achieving useless metrics. Further, with new software testing tools and procedures developed into Microsoft’s Visual Studio programming platform, the company decided to combine the responsibilities of testing and development in testing into the singular developer role.

So now, everyone is a developer, and everyone writes test cases and tests it themselves. It translates into a better product, it makes us more efficient in the right way.45

Another manager explained that this allowed the firm to hire better developers.

We can go to any university and hire, or hire people with experience ... people that can just jump right in and don’t require a lot of hand holding.46

This describes the use of automation by Microsoft within its workforce. Microsoft created Visual Studio for other firms to develop on its platform, but it was also developed to automate low-skilled labor—which would appear in the coders or programmers classifications

44Seattle principal at a large firm. Interview, 11/20/2015, D30, transcript.
45Seattle principal at a large firm. Interview, 11/20/2015, D30, transcript.
46Seattle product manager at a large firm. Interview, 10/01/2016, D23, transcript.
and would incorporate a tool-dependent, high-skilled version into the already high-level engineering positions.

As I showed earlier in the chapter, workers go between Microsoft and Amazon.

Workers go back and forth all the time. So when they do something new, it’s only a matter of time before it makes its way over, and same goes if we do something new. ... We’re different companies, we do things different and we use different technologies, but there’s a lot of spillover ... in how things get done.47

Portland and Austin are similar in their occupation structures, but Portland has fewer high-level engineers and more coders and programmers. In Portland coders and programmers comprise one-third of the software work force, yet they comprise less than one-quarter in Seattle. In my Portland interviews, two key variables became evident in describing Portland’s tech scene: open source and a networked community.

You don’t even have to try hard. Just show up [to a meetup] and someone will have you coding .. they’re all on Calagator [Portland-developed meetup and user group tracker48]. It’s easy to get started.49

Another Portland developer and informal mentor at multiple meetups placed importance on the combination of open source and local community.

I give people tasks on this little project I’ve been working on. It’s silly. But it’s free [open source], and then they interact with people all over the world, which is really empowering, especially when they come back to [the meetup] and talk about it.50

---

47 Seattle developer at a large firm. Interview, 10/19/2016, D26, transcript.
48 Calagator: Portland’s Tech Calendar (http://calagator.org/).
49 Portland developer at an SME. Interview, 10/20/2015, D16, transcript.
50 Portland developer at an external SME. Interview, 09/10/2015, D32, notes.
The developer continued to explain that their role as a mentor meant staying alert for new jobs and teaching mentees how to prioritize meetups and events in the region. Open source, they explained, made it so that anyone could learn to develop code quickly. Cloud-infrastructure tools – like those provided by Microsoft and Amazon – give new developers the ability to scale their programs and focus on improving their coding skills without having to worry about servers and operations.

Austin is similar to Portland, but software workers are still establishing an open-source identity. Further, the community centers around Capital Factory and is not dispersed among firms or coding schools, like in Portland. Capital Factory places importance on venture capital, a different model than Portland, and software workers in Austin work for the state government either directly or through the University of Texas. The larger number of small firms in Portland and Austin, combined with the desire to hire workers that don’t require hand-holding in Seattle, contributes to the shares of managers in each region. Regions with more small- and medium-sized firms require more managers, while those with large firms seeking to reduce the need for managers will have a smaller share of managers.

5.5 Conclusion

This chapter sought to understand the local nature of software production and the development of software agglomerations in Seattle, Portland, and Austin. I asked two specific questions:

1. How does software production activity vary between region?

2. What are the industrial district formations of software production in Seattle, Washington, Portland, Oregon, and Austin, Texas?
Figure 5.6: Software occupation share of region labor force, stacked, ACS 2011–2015.

Author’s calculations from data gathered from IPUMS-USA (Ruggles, Genadek, et al. 2015) ACS 2011–2015. Software occupations include OCC codes 110, 1005, 1006, 1007, 1010, 1020, 1030, 1050, 1060, 1105, 1106, 1107, 1200, 1220.
3. What are the regional outcomes, via occupational structures and income distribution of software workers, of regionally specific modes of software production?

To address the first question, I leveraged the IPUMS-USA data source from Chapter 4 and used a geographic weighting system to construct a comparable metro dataset of software workers between 1970 and 2015. I traced the changing fortune of US metros’ software agglomerations and showed how the historical development of software co-evolved with spaces of software production. I then constructed industrial district typologies based on interviews, historical research, and participant-observation, and using the characteristics of industrial districts in Table 2.1, Table 2.2, Table 2.3, and Table 5.3. I then combined this data with participant-observation and interviews in the third section to understand the development and variation between the three software agglomerations of Seattle, Portland, and Austin.

To address the third question, I again referred to those three case regions. For each region, I demonstrated how historical developments resulted in software industrial districts with regionally specific modes of production. I examined the history of the three case regions, their industrial districts, and the ways in which their software firms and workers produce software. Specifically, I analyzed each region’s forces of production (technologies, skill-sets, labor markets), social relations (class-based firm-worker relationships), and spatial organizations (industrial district formations). With this data, I created industrial district types to reflect my three cases regions and detailed their industrial composition. I used Table 5.3 to add software-production-specific characteristics to the idealized industrial district typology. Then in the last section, I showed how these regionally specific modes of software production, and their differences, yielded local occupation structures and income distributions.

Further, I argued that these second-tier cities challenge the dominance of Silicon
Valley’s software agglomeration. And that the fates of regional software agglomerations lies in the ability of the firms and workers in the industrial districts to adapt to the new profit cycles of new software periods. For example, Microsoft’s dominance in the mass-market software period did not ensure success in the cloud-and-mobile software period, but with new leadership, Microsoft is starting to adapt. Amazon emerged as a cloud-and-mobile software firm and may have helped build regional specialization in cloud and mobile technologies. Similarly, Portland and Austin had small software agglomerations that emerged from different historical trajectories in the 2000s. Both regions benefited from adjacent industries, and Portland benefited from the development of an open-source network of software workers across industries. Lastly, in Section 5.3, I show how the low-materiality of software, and the requirement to use software to produce software, afforded the evolution in software’s capitalist production variants. In this chapter, I created and operationalized a framework that connected the periods of software to profit cycles, industrial districts, and capitalist variants in software production.
6 Segmentation and Bifurcation: Equity and Economic Development

[T]he very shortage of programmers and their resulting privileged status is giving strong impetus to the development of programming tools that could de-skill programmers or reduce the demand for them substantially.

Hall, Markusen, Osborn, and Wachsman 1983, 36

The workforce in these industries is increasingly coming to be divided between highly qualified scientific and technical staff and semiskilled (predominantly female) assembly workers. This dichotomisation of skill within the labour force has some tendency to be reflected within the spatial pattern of the industry.

Massey and Meegan 1978, 283

Profound discontinuities in the labor market are legitimized through the attribution of skilled status. In this sense, skill should not only be seen simply as a resource that is rewarded in accordance with the precepts of human capital theory, but as an ideological construct reflecting the distribution of power in the labor market.

Peck 1996, 135

6.1 Introduction

The seemingly always newness of software production, and high-tech in general, may afford a conceptual disconnect between a region’s software production and the existing
social and spatial relations of production. But, as I’ve shown in Austin, Portland, and Seattle, software production transforms between each software period, taking on place specific modes of production that evolve out of a region’s existing firms, industries, people, and their connections. That is, as software production globally evolves, the local specificities of social and spatial relations act as scaffolding in the new periods that configure local software production. As we saw in Portland and Austin, for example, their hardware orientation in the 1970s and 1980s during the corporate software period meant that the regions missed much of the benefit of the software commodity period, but are now benefiting from business to business cloud software (Portland) and client-side cloud software (Austin). Software production activity is a continuation of a regions specialized knowledge, arranged by its industrial district.

In chapter 4, I showed how the historical development of computing and software yielded a labor market structure that spans industries and, because of labor rationalization, whose internal logics began to mimic the professionalization, forms of skillbuilding, and certification of electrical engineering occupations. I linked the evolution software production to different forms of capitalist production and their multi-decade profit cycles. Within each of software’s capitalist variant of production and associated profit cycles, I showed how software workers, and the IT departments they work under, frequently undergo restructuring due to the diffuse nature of software work, the perpetual desire for scientific management, and the constant churn of technological (software) development and (software) automation.

In the last chapter, chapter 5, I showed that the restructuring of software production over four periods varies by region. Regional modes of production – sets of social and spatial relations determined by firm strategies, labor market practices, and technologies – yield variation in occupational structures. Importantly, the occupational structures determine the distribution of occupations, power relationships between occupations, and
the income distribution in the region.

In this chapter, I connect the resulting occupational bifurcation to firms exercising capitalist logics of the detailed and spatial divisions of labor. Technological development, in the pursuit of profit, drive the transition of profit cycles and development of new forms of capitalist relationships in each software period. Within the realm of production, the resulting software developments automate existing forms of production, provide increased efficiencies where software is already used, and afford new forms technical possibility. For workers, the result worker deskilling and continued alienation, existing occupations become bifurcated, and reskilling is necessary to tap into new technical possibilities. These outcomes feed back into the cycle, necessitating that firms seek more automation through software development as firm management may seek to divide the individual components in production in pursuit of efficiency.

Occupational bifurcation manifests as labor market dualization and leads to locally specific forms of segmentation. Labor market segmentation – the development of sub-labor market(s) divided by skill, gender, or race & ethnicity – builds from local social and spatial configurations resulting in a high-skill labor market and a low-skill labor market. The low-skill software labor market is distinctly more female and more non-white, raising questions of career mobility and equity. Privileged workers, when threatened by deskilling or decreases in occupational status and wages, benefit from institutional forms of sexism and racism and tend to maintain their status and/or wages. Women and minorities become disadvantaged in the labor market. Spatially, firms may separate design from production, choosing to locate an engineering department in one city and a support team in another. Depending on the mode of production, the type and method of producing software, different sets of occupations – with real wage differences – may concentrate in certain regions.

Addressing these issues, and building on the last chapter where I showed that regionally
specific modes of production yield different regional outcomes through locally specific occupational structures, this chapter asks:

1. To what extent has software production demonstrated tendencies toward labor market dualization and segmentation by gender and race/ethnicity?

2. How are these labor market outcomes mediated by local institutions and practices?

I answer the first question in the second and third sections. First, I further examine the dynamics of occupation bifurcation and dualization of the labor market. In Tables 6.1 and Figure 6.1, I present the historical trends of bifurcation in the software occupations. In Table B.1, in the Appendix, I collect occupation descriptions and national statistics on for each occupation, including incomes, income breakdowns by gender and race/ethnicity, and the number of workers. Based on function and incomes, I further group occupations into four tiers of the labor market. I use these four tiers to understand gender and race/ethnicity dynamics, addressing the feminization of the labor market.

In the fourth section I address the second question. I leverage both participant-observation and interviews to inform the local responses to labor market outcomes. I show how local institutions, organizations, and practices provide different leverage points to counter the uneven impacts of labor market dualization and segmentation.

In the conclusion, I combine these three sections. Labor market dualization and segmentation are broad structural forces, but I introduce both software-specific and locally-specific tendencies. I argue that:

1. The product of software labor is software, and software is the same “machine” used to automate software work. The rapid pace of software development thus coincides with the rapid deskilling of software workers. Software workers thus
implicate themselves in creating programs which deskill themselves and other software workers.

2. Regionally specific modes of production mold the dualization and segmentation of the local software labor market. Nationally, the share and number of women and persons of marginalized racial/ethnic backgrounds is growing, but primarily at the lowest tier of the labor market. Occupations in the lowest tier are more precarious than their higher-skilled counterparts, require less skill and provide less income. Local variations on national trends depend on the regional mode of production.

3. Labor market intermediaries — through initiatives and groups within large firms like the example in Seattle, through community-led events like the example in Portland, or through community-organizations like the example in Austin — play an important role in enabling marginalized workers to build technical skills and gain soft, or tacit, skills to better navigate the labor market.

This chapter argues that regional divergence in software production shape the form of, and the responses to, labor market dualization and segmentation. I show that labor market dualization and segmentation manifest differently in each region. I show that local responses, mediated by local institutions and local practices, are necessary to mitigate the detrimental impacts of the local manifestations of segmentation and bifurcation.

6.2 Bifurcation: The Race to Stay Competitive

6.2.1 The firm logic to divide tasks

In the 1980s, Hall, Markusen, Osborn, and Wachsman research the computer industry and note how software is different than computer manufacturing. Their epigraph at the beginning of this chapter is a reminder, that, even in the 1980s with a small overall labor
force, there was strong desire to reduce management’s reliance on software workers. I include the epigraph below, but with more context.

... present day programmers come, for the most part, from a small, well-educated, professional segment of society with rather specialized aptitudes... and the very shortage of programmers and their resulting privileged status is giving strong impetus to the development of programming tools that could de-skill programmers or reduce the demand for them substantially. (Hall, Markusen, et al. 1983, 36)

Judging from the growth in the size of the software labor market, demand for software workers, has increased, intensifying pressure to deskill or reduce the demand, and reliance, on software workers.

To reduce demand on software workers, or at least attempt to mitigate its impact, managers document software work and split it into subcomponents, bifurcating each individual occupation into higher-level cognitive tasks and lower-level repetitive tasks. Not all occupation splits reflect this split, as I’ll get to later, but the pattern of splitting between high- and low-level tasks is pervasive in software work.

Yeah. They split us into front-end and back-end. I started on front-end ... but now I do back-end. It’s more rewarding.51

Another interviewer put the dichotomy between the two types of positions into perspective.

Usually I ask myself if we can just leverage a library to do what a front-ender does. Can we bootstrap it?52

51Seattle developer at a large firm. Interview, 10/10/2016, D27, transcript.
52Portland developer at an SME. Interview, 11/21/2016, D28, transcript.
Splitting work between front-end and back-end on the web may, at first, appear to be natural. There are alternate ways to split the work, but today, because of the advent of high-level web-based languages and libraries, front-end development work has less prestige than back-end development work. Back-end work has evolved too, but not in the same way that front-end work has evolved into high-level software activity. The development of high-level code is both a way to automate and to gain new capabilities. This dialectic is not lost on software workers.

Low-level code is for really specialized purposes, it what you use to make the high-level possible. High-level code is democratizing. Yes, it’s automation, but the idea is that you can give everyone the ability to write high-level code and you only need a few low-level people. Think what VB [Visual Basic] did – it’s mind blowing.53

Visual Basic (VB) – introduced by Microsoft in 1991 as an event-driven, third-generation programming language – introduced new efficiency among office workers. VB has simple instructions and integrates into the entire Microsoft suite of applications. The net gain in efficiency, measured relative to other firms, becomes negligible because of the near universal adoption and integration into Microsoft’s popular office suite. On the one hand, languages like VB deskill some of the work that other programmers do by making programming more accessible, but it also requires that VB receives wide adoption through skill-building. Users of Microsoft Excel use VB functions without realizing the underlying technologies.

The software workers I interviewed all generally shared the opinion that more people, if not everyone, should know how to program. Interviewees shared the idea that technology was advancing, without identifying the drivers of technological development nor...
identifying the broader social and technical issues for which software is developed, or should be developed, to resolve. One interviewee, in particular, was aware of the threat of becoming less competitive, expressing his need to be reading the newest blog posts on server-side efficiency and learning new languages.

I had to learn puppet [programming language] or else ... or else I would become a dinosuar [hesitant laughter].

Two of my interviewees described the need to stay updated in new languages and libraries as a “race condition.” A race condition refers to the problematic program execution situation where the output of a program depends on the execution sequence of two or more other underlying subprocesses. A race condition may produce undesirable and unexpected results, or crash the program entirely, when the underlying subprocesses run in a sequence not intended by the programmer either through lack of specification or events that the programmer cannot control. The interviewees expressed anxiety over feeling pressured to race, literally, the constantly changing knowledge requirements of their work with the threat of undesirable or unforeseeable implications if they did not stay up to date.

All of the interviewees mentioned the importance of more people, if not everyone, learning how to program and develop software. Some interviewers went as far as to say that learning to program was part of a natural evolution, like learning how to read and write. Technological development, including software development, does happen through experimentation. Firms, led by their managers, invest time and money to increase efficiency by reducing their demand on labor, either adopting software which replaces non-software workers or by adopting libraries which reduce or deskill existing

---

54 Portland developer at an SME. Interview, 05/05/2016, D31, notes.
55 Portland developer at an SME. Interview, 05/05/2016, D31, notes.
56 Austin developer and entrepreneur. Interview, 10/27/2015, D4, transcript.
software workers. Between 1970 and 1990, software workers were not managed by a manager specific to software or information technology. My interviewees that were programming in this period, and the literature, suggest that software workers instead reported directly to business managers, C-level executives, or their direct reports (Reskin and Roos 1990; Greenbaum 1979). Managers, even those with limited understanding, sought to breakdown the tasks of software development to shorten development cycles or increase development output.

Quotes in previous chapters showed how managers quarantine developers from interaction with other business units so that they have the most time to develop software. Managers expect software workers to already have a skill-set and use other software programs to increase their efficiency. Managers also expect their developers to keep scrupulous notes, or code comments, to document not only what the program does, but method of production. Teams meet to break down the tasks and assign difficulty, time, and a team member to each task – ultimately making the software worker replaceable.

Returning back to Hall, Markusen, et al., the business goal is to ...

... reduce the number of people required to produce the same number of units of software and/or to transfer as much work as possible from highly-trained (expensive) to lesser-trained (cheap) personnel. One model for this is the ‘software factory,’ a hierarchical, assembly-line in which software production is broken down into small stepwise tasks which are performed by less-skilled programmers passing their little pieces from cubicle to cubicle. (Hall, Markusen, et al. 1983, 34)

The need for software workers to stay relevant is in part connected to the ways firms need to manage, and reduce, costs. Firms do so by using software tools which reduce the need, or quantity, of workers necessary to complete tasks that would otherwise require
specialized skill sets and thus expensive workers. Software firms are susceptible or prone to adopt productivity or automation tools in software development. This allows for the firm to hire less expensive *users* of software and reduce the number of *developers* of software – hence, bifurcating the occupation, and creating a dual labor market. As software workers introduce new software libraries and languages, software workers themselves race to stay competitive on the labor market by learning new skills. As detailed in the last chapter, interviewees in Austin and Portland expressed how new software is vital to themselves as software workers, but also to the success of the local software community. We saw in Portland that the rise of open source helped its software agglomeration grow. In Austin, ruby and python frameworks are key to its success.

### 6.2.2 Five changes in the software occupational structure

The Bureau of Labor Statistics (BLS) and the US Census Bureau collect information about occupations and group them according to function and salary. Importantly, they develop an occupational classification system that represents the different types of occupations across industries – that is, the classification system isn’t a prescription and is instead a reflection of the types of work that workers perform. The period between 1990 and 2000, the largest growth in software labor of any period in this study, present three noteworthy phenomenon in this regard.

First, a new managerial class dedicated to overseeing software work emerged (occupation 110 in 1990, see Figure 6.1). Computer and information systems managers, its formal title, is not a recode of other managerial positions (less than 5% come from other managerial occupations) and is not a recode of existing software workers, the occupation is new (Beckhusen 2016). The emergence of this new class validates the business use of software and the need to have specialized knowledge about how manage technical projects, whether adopting and integrating existing software or building software within
the firm, and the software workers assigned to those projects. Firms even develop entire departments whose tasks are technical business operations, employing analysts and administrators.

Second, the computer systems analyst and scientists occupation splits into two topically different occupations, representing the emergence of the internet, a network focused analyst position and a non-network focused analyst position. This, unlike the splits mentioned later, does not necessarily represent the bifurcation of an occupation, but the topical differentiation between those working on software for desktops and mainframe servers versus software for the internet, firm intranets, and web servers.

Third, the computer programmer position splits into software engineers and computer programmers. This marks the first bifurcation of software occupations that existing data make available to track. Other bifurcations happen in the 1970s but are not documented in occupation codes (Reskin and Roos 1990, see for example). The computer programmer in 1990 (occupation 229 in Figure 6.1) had a median annual salary $60k per year. In 2000, software engineers (later software developers), the top-tier of the occupational split had a median annual salary $87k and there were fewer women and a higher wage gap between genders and between white non-Hispanic workers than and their minority counterparts. The lower-skilled of the split, in the middle-tier labor market, also saw in increase in yearly salary, but at only a fraction of the increase of their higher-skilled counterparts. The computer programmer earned $72k per year in 2000, and had higher shares of women, minorities, and a smaller wage gap than the computer programmer in 1990. See occupations 1020 (software engineers/developers) and 1010 (computer programmers) in Table B.1. When the occupation splits according to the logic of management, the lower skilled occupation takes on a routinized role while the higher skilled occupation takes on a higher-skilled role. Generally, fewer higher-skilled positions exist where there were once

57Excludes Asian non-Hispanic, and future references to minority in this section.
lower-skilled positions.

These first three moments – one new occupation, a topical split, and a skill-based split – have ramifications today. The management role continues to be the highest paid software occupation and continues to grow. The topical split of computer analysts into network-based and non-network based set the ground work for another two splits between the 2010 occupation codes and the occupation codes used for today (evident between the 2005-2009 period and the 2011–2015 period). And, since their equal split in 1990-2000, the number of workers in a software engineer/developer role has double while the number of workers in the computer programmer occupation has decreased slightly, indicating the rapid and continued expansion of technical occupations. The last split, an occupational bifurcation, in 1990 represents the continued growth of network and internet technologies. The two resulting occupations from the computer systems analysts and scientists occupation both split again in 2005-2009 making evident their capture and codification of these two occupations by management and the development of software libraries and languages.

The fourth and fifth splits happen between 2005-2009 and 2011–2015. The fourth split, another bifurcation, divides the computer scientist and system analyst (occupation 1000 in Table 6.1) into research scientists, systems analysts, and all other computer occupations. During this time period, workers in these occupations grew from 728,000 to 920,000. Prior to the recoding, the position of computer scientist and computer analysts were nearly indistinguishable as both required technical skill to analyze computer models and automate systems. However, with new tools to automate analysis, and in fitting in line with turning software production activity into software using activity, a break appeared in the reported occupations: computer systems analyst was its own separate activity. Prior to the break, the joint computer scientist and systems analyst earned a median annual salary of $70k per year. After the split, the 13,000 workers in the higher-level occupation
had a median annual salary of $74k, had fewer women and minorities and a larger wage gap for both women and minorities. As expected, the lower-level computer systems analyst retained 490,000 workers, had median annual salary of $71k, had more women and minorities, and had less of a wage gap than the research scientist counter part. The all other computer occupations category is difficult to track because of the lack of clarity in the varying functions. All other computer occupations had 418,000 workers, had the lowest salary, highest rates of women and minority participation, and the smallest wage gap compared to the two other occupations. These numbers are presented in Table B.1 for computer scientist and system analyst (occupation 1005), computer systems analyst (occupation 1006), and all other computer occupations (occupation 1107).

The fifth split, also a bifurcation, is the split of the network systems and data communication analysts (occupation 1110 in Table 6.1) into computer network architects, information security analysts, and web developers. This is the clearest example of the bifurcation of an occupation into new distinct occupations with their own labor markets.

In 2005-2009, the network systems and data communication analyst had a median income of $60k per year, was nearly a fifth women and there was near wage equity between white workers and their minority counterparts. When the occupation split, it

Figure 6.1: (Next page) Author’s representation using data from Beckhusen (2016), the Bureau of Labor Statistics Standard Occupation Classification (Labor Statistics 2010), the IPUMS-USA Occupation Crosswalks (Ruggles, Genadek, et al. 2015).

Figure 6.1: Significant occupation recoding of software labor by occupation code, 1970–2015.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-2009</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011-2015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 Comp &amp; info sys managers (303,238)</td>
<td>11 Comp &amp; info sys managers (423,651)</td>
<td>11 Comp &amp; info sys managers (524,707)</td>
<td>10 Comp &amp; info sys managers (524,707)</td>
<td>10 Comp &amp; info sys managers (524,707)</td>
<td>10 Comp &amp; info sys managers (524,707)</td>
<td></td>
</tr>
<tr>
<td>10 Comp sci &amp; sys analysts (653,110)</td>
<td>10 Comp sci &amp; sys analysts (728,294)</td>
<td>10 Comp sci &amp; sys analysts (803,378)</td>
<td>10 Comp sci &amp; sys analysts (878,562)</td>
<td>10 Comp sci &amp; sys analysts (953,746)</td>
<td>10 Comp sci &amp; sys analysts (1,028,930)</td>
<td></td>
</tr>
<tr>
<td>10 Network &amp; comp sys admins (213,801)</td>
<td>10 Network &amp; comp sys admins (283,283)</td>
<td>10 Network &amp; comp sys admins (352,765)</td>
<td>10 Network &amp; comp sys admins (422,247)</td>
<td>10 Network &amp; comp sys admins (491,729)</td>
<td>10 Network &amp; comp sys admins (561,211)</td>
<td></td>
</tr>
<tr>
<td>10 Database admins (95,138)</td>
<td>10 Database admins (165,020)</td>
<td>10 Database admins (234,902)</td>
<td>10 Database admins (304,784)</td>
<td>10 Database admins (374,666)</td>
<td>10 Database admins (444,548)</td>
<td></td>
</tr>
<tr>
<td>10 Comp software engineers (436,159)</td>
<td>10 Comp software engineers (506,360)</td>
<td>10 Comp software engineers (576,561)</td>
<td>10 Comp software engineers (646,752)</td>
<td>10 Comp software engineers (716,943)</td>
<td>10 Comp software engineers (787,134)</td>
<td></td>
</tr>
<tr>
<td>10 Comp programmers (566,709)</td>
<td>10 Comp programmers (636,019)</td>
<td>10 Comp programmers (705,330)</td>
<td>10 Comp programmers (774,641)</td>
<td>10 Comp programmers (843,952)</td>
<td>10 Comp programmers (913,263)</td>
<td></td>
</tr>
<tr>
<td>10 Ops research analysts (66,990)</td>
<td>10 Ops research analysts (75,341)</td>
<td>10 Ops research analysts (83,692)</td>
<td>10 Ops research analysts (91,043)</td>
<td>10 Ops research analysts (98,394)</td>
<td>10 Ops research analysts (105,746)</td>
<td></td>
</tr>
<tr>
<td>10 Network &amp; data comms analysts (335,975)</td>
<td>10 Network &amp; data comms analysts (396,237)</td>
<td>10 Network &amp; data comms analysts (456,500)</td>
<td>10 Network &amp; data comms analysts (516,763)</td>
<td>10 Network &amp; data comms analysts (577,026)</td>
<td>10 Network &amp; data comms analysts (637,292)</td>
<td></td>
</tr>
<tr>
<td>10 Web devs (188,937)</td>
<td>10 Web devs (259,399)</td>
<td>10 Web devs (330,861)</td>
<td>10 Web devs (402,323)</td>
<td>10 Web devs (473,784)</td>
<td>10 Web devs (545,246)</td>
<td></td>
</tr>
<tr>
<td>10 Software devs, all (1,036,507)</td>
<td>10 Software devs, all (1,107,969)</td>
<td>10 Software devs, all (1,179,431)</td>
<td>10 Software devs, all (1,250,903)</td>
<td>10 Software devs, all (1,322,374)</td>
<td>10 Software devs, all (1,393,846)</td>
<td></td>
</tr>
<tr>
<td>10 Comp programmers (490,119)</td>
<td>10 Comp programmers (559,470)</td>
<td>10 Comp programmers (628,821)</td>
<td>10 Comp programmers (698,172)</td>
<td>10 Comp programmers (767,523)</td>
<td>10 Comp programmers (836,874)</td>
<td></td>
</tr>
<tr>
<td>10 Ops research analysts (439,936)</td>
<td>10 Ops research analysts (506,360)</td>
<td>10 Ops research analysts (572,781)</td>
<td>10 Ops research analysts (639,202)</td>
<td>10 Ops research analysts (705,623)</td>
<td>10 Ops research analysts (772,044)</td>
<td></td>
</tr>
<tr>
<td>10 Network &amp; computer sys admins (205,491)</td>
<td>10 Network &amp; computer sys admins (275,953)</td>
<td>10 Network &amp; computer sys admins (346,415)</td>
<td>10 Network &amp; computer sys admins (416,877)</td>
<td>10 Network &amp; computer sys admins (487,339)</td>
<td>10 Network &amp; computer sys admins (557,801)</td>
<td></td>
</tr>
<tr>
<td>10 Database admins (107,724)</td>
<td>10 Database admins (178,186)</td>
<td>10 Database admins (248,648)</td>
<td>10 Database admins (319,110)</td>
<td>10 Database admins (389,572)</td>
<td>10 Database admins (459,034)</td>
<td></td>
</tr>
<tr>
<td>10 Comp support specialists (566,709)</td>
<td>10 Comp support specialists (636,019)</td>
<td>10 Comp support specialists (705,330)</td>
<td>10 Comp support specialists (774,641)</td>
<td>10 Comp support specialists (843,952)</td>
<td>10 Comp support specialists (913,263)</td>
<td></td>
</tr>
<tr>
<td>10 Software devs, all (1,036,507)</td>
<td>10 Software devs, all (1,107,969)</td>
<td>10 Software devs, all (1,179,431)</td>
<td>10 Software devs, all (1,250,903)</td>
<td>10 Software devs, all (1,322,374)</td>
<td>10 Software devs, all (1,393,846)</td>
<td></td>
</tr>
<tr>
<td>10 Comp programmers (439,936)</td>
<td>10 Comp programmers (506,360)</td>
<td>10 Comp programmers (572,781)</td>
<td>10 Comp programmers (639,202)</td>
<td>10 Comp programmers (705,623)</td>
<td>10 Comp programmers (772,044)</td>
<td></td>
</tr>
<tr>
<td>10 Ops research analysts (75,341)</td>
<td>10 Ops research analysts (83,692)</td>
<td>10 Ops research analysts (91,043)</td>
<td>10 Ops research analysts (98,394)</td>
<td>10 Ops research analysts (105,746)</td>
<td>10 Ops research analysts (113,101)</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Bifurcation of software labor, 1970 to 2015.
split into two separate, but unequal, top-tier high-skill positions and a bottom-tier low-skill occupation that is the lowest paid occupation of all software work. The 2011–2015 breakdown is as follows. The top-tier high-skilled network architect occupation (see occupation 1106 in Table 5.9 and Table B.1) had a median annual income of $90k and had a small wage gap for women, but lost the majority of the women in the occupation and women comprised 10% of the occupation. Minority representation stayed high and wage gap increased only slightly. The ratio of median incomes between minorities to white, non-Hispanics was 1:1 in 2005-2009 and 0.82:1 in 2011–2015. The top-tier high-skilled information security analyst (see occupation 1007 in Table 5.9) had a median annual income of $80k and had larger shares of women and minorities than the original position. Income disparities increased to 0.82:1 for both women and minorities. This position, while as prestigious as the network architect occupation, remains in the top-tier labor market with its high annual salary and degree requirements. Most alarming is the third occupation in the low-tier low-skill labor market. Web developers (see occupation 1030 in Table 5.9) had a median annual income of $42k. The occupation maintains about 15% minority representation and increases representation of women from 25% to 35%. The drop in median income is met with decreases in the wage gap of other two top-tier occupations, dropping slightly to 0.84:1 for women and decreasing for minorities to 0.99:1.

Changes reported to American Community Survey, and to some extent the Bureau of Labor Statistics, result in the occupational recoding. They approximate the functional and occupational changes within firms. From discussion with my interview subjects and observations job hunting sites, the web developer position is not new to the 2011–2015 period, suggesting a delay between initial observation and its normalization into an occupation. Nonetheless, the topical split of analysts in 1990 into a network-based occupation led the pathway for the split of the network systems and data communication analyst
into three occupations in the 2011–2015. While all three occupations produce software — whether designing cloud infrastructure, monitoring and securing cloud infrastructure, or developing applications for cloud infrastructure — the key differentiator between the three positions is the degree in which workers in these occupations use software. The description provided by the Bureau of Labor Statistics for the web developer occupation even includes the phrase “may use software” which does not appear in the descriptions of any of the other occupations. The other bottom-tier low-skilled occupation, computer support specialist, earns on average more than their web developer counterpart and using software is part of that occupation, but not to the degree of the of web developer.

The codification of one part of a network systems and data communication analyst into occupation is at the heart of bifurcation in software production activity. The web developer roles demonstrates the use of software and management rationalization to decrease the labor operations cost of the firm. The relative higher wage ratios for women and minorities to their comparator groups is an artifact of the compressed wage rage as the mean income of the web developer, again, is only $42k per year. The occupation still generally requires a 4-yr degree and training.

6.2.3 Bifuraction toward what end?

In this section, I used interview quotes and five moments in the occupational restructuring of software work to show that with the growth of software production has come the need for firms to reduce their labor costs by using software to replace developers with users. Within software, this might happen when a software worker uses a library, a higher level language, or a machine-learning techniques to to gain programmatic efficiency, displacing the work of other software workers. The product of software labor is software, and software is the same “machine” used to automate software work. The rapid pace of software development coincides with the rapid deskilling of software workers — what my interview
subjects expressed as a an anxiety-ridden “race condition.” Software workers therefore implicate themselves in creating programs which deskill themselves and other software workers. Lastly, I showed that from the direction of managers and the codification of workers actions, occupations undergo splits, ultimately bifurcating occupations.

In the next section, I dig deeper into the impacts on women and minorities through the creation of dual labor markets within the broader software labor market.

6.3 Segmentation: Sub-labor markets, race, and gender

6.3.1 Toward feminization

One of the occupational splits was topical in nature, but the other three splits exemplify occupational bifurcation. The process of occupational bifurcation creates dual labor markets. As software occupations bifurcate into two or more occupations, with at least one top-tier high-skilled position and at least one lower-skilled position in the middle-tier or bottom-tier, the bottom of the software labor market grows. This lowers the barrier to entry within software as a whole. Importantly, as the low-end of the labor market grows, because of the resorting by employers of the labor queue, the bottom of the labor market begins to include those who may have not had access to these jobs: primarily women and persons of color. Yet, this new job is not the same as the job that their primarily white and primarily male counterparts had and promotion in this already white and male dominated set of occupations is often dependent navigating ideals of masculinity.

I present national level detail of the software labor force on median income and labor force representation by gender in Figure 6.2 and Table 6.2. Similarly, I present the same type of data by race/ethnicity in Figure 6.3 and Table 6.3. I've separated Asian non-Hispanic (Asian) and white non-Hispanic (white), creating a minority group that summarizes all other race/ethnicity combinations.
Table 6.2: National median income by gender for software workers, 1970-2015

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of workforce</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Men</td>
<td>240,700</td>
<td>394,380</td>
<td>858,111</td>
<td>2,347,303</td>
<td>2,662,948</td>
<td>3,156,591</td>
</tr>
<tr>
<td>Women</td>
<td>50,300</td>
<td>145,580</td>
<td>430,873</td>
<td>947,094</td>
<td>955,347</td>
<td>1,088,380</td>
</tr>
<tr>
<td>Percent women</td>
<td>17%</td>
<td>27%</td>
<td>33%</td>
<td>29%</td>
<td>26%</td>
<td>26%</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Men</td>
<td>71,390</td>
<td>66,538</td>
<td>70,122</td>
<td>75,362</td>
<td>77,893</td>
<td>75,879</td>
</tr>
<tr>
<td>Women</td>
<td>46,387</td>
<td>46,582</td>
<td>58,435</td>
<td>62,174</td>
<td>65,561</td>
<td>64,074</td>
</tr>
<tr>
<td>Income ratio</td>
<td>0.65</td>
<td>0.70</td>
<td>0.83</td>
<td>0.83</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All software workers</td>
<td>66,126</td>
<td>166,304</td>
<td>35,460</td>
<td>48,116</td>
<td>74,220</td>
<td>72,262</td>
</tr>
</tbody>
</table>


This data suggests that minorities and women receive only part of the pay that of their white and male counterparts. Further, the concentration of women and minorities toward the bottom of the pay scale suggest that they are in the bottom of the labor market. Minority representation had quadrupled since 1970. The wage gap for minorities decreased between 1970 and 1990, then increased drastically 1990 and 2000, before flattening. The wage gap for women has only improved since 1970. However, there has been little change for either minorities or women since 2000 as both have had the same wage ratio of about .84 or .83. Women’s participation in software occupations increased between 1970 and 1990, reaching 30%, but since then has only decreased to about 26%.

This data shows that during the mass-market software period of the 1990s and early 2000s, gains by women and minorities slowed or even reversed. This also coincides with the largest growth of software workers, suggesting that as software work grew and
software production became a more desirable job, white men and Asian, workers possibly recruited internationally, filled these vacancies at a greater rate than women and minorities. Between no period does the raw number of any group decline.

The income violin plots – with box plots to show standard deviation and an asterisk to show median income for each group – tell many stories. First, women and minorities entering the software labor market enter, or stay, at the bottom of the income distribution at greater rates than men. Second, the high number at the bottom suggest that there are fewer opportunities for advancement, or that women leave the workforce earlier. There are also fewer women at the top of the income distribution when comparing to men. For example, at the top of the income distribution, 4.3% of men make over $150k per year while 1.7% of women cross that threshold. To compare, if we were to find the income that separates the top 4.3% of women (effectively the 95.7th percentile), the annual income drops to $122k per year. The ratio of the top 4.3% salary of women to men is 0.81:1, a higher wage gap than the median incomes of women to men, evidence of higher gender inequality in higher income software occupations.

The percentages in themselves are staggering, but so are the raw numbers: 4.3% of men software workers is 412,963 men, while 1.7% of women software workers is only 63,279 women. Certainly social reproduction plays an important part in the income distribution because of career tenure and career interruptions like maternity leave. However, the differences in incomes, income distributions, and labor market representations suggest structural issues related to bifurcation of occupations and the segmentation of the labor market.

The income violin by race/ethnicity shows similar results. The relationship between the income distribution of white software workers and minority software workers mimics that of the relationship between men and women software workers above. Minority software workers experience higher income inequality at higher income levels. The
The structural shaping of the software labor market – through occupational bifurcation and the dualization of the labor market – isn’t exclusively a recent phenomenon. Prior to the significant gains of both women and minorities leading up to 1990, Reskin and Roos document the feminization of software production and computer work in
the 1970s. They show that a variety of factors led to the increased role women in the workforce.

...as the computer workforce grew, managers sought to contain labor costs by separating from programming two new occupations: systems analysts, who designed information systems; and coders, who translated programs into computer codes and entered data. (Reskin and Roos 1990, 43)

This split happened in the US prior to the beginning of this study. The women in their study who entered the workforce were not able to enjoy the same benefits as their male counterparts. Unlike the other industries that Reskin and Roos document, new women entrants into software production experienced ...

... segregation ... between mostly female low-paid analysts whose daily tasks involve interacting with users and mostly male high-paid analysts who make
decisions about information systems and interact with analysts, programmers, and managers. (Reskin and Roos 1990, 182)

The bifurcation of software occupations in the last section has the same makings of the feminization that Reskin and Roos identify. This structural inequity between gender and race/ethnicity reproduces itself, further segregating women and minorities. Feminization suggests a growing number of women and minorities in the work force, but they show, and this data reinforces, the growth is in the lower end of the labor market. Today, this garners greater importance because since the work of Reskin and Roos, the software production labor market in the US has grown tenfold (1980), the share of women in the software workforce has steadily decreased (1990), and wage gap of minorities has steadily increased (1990).

In Figure 6.4, Figure 6.5, and Table 6.4, I show income distributions for the three case cities by gender and race/ethnicity. I highlight the regional differentiation between regions, due to their mode of production. In line with the national datasets, we see that the income distributions for women and minorities are at lower incomes than the income distributions for men and whites.

Austin’s mixed industrial district typology, exhibiting both Marshallian and public-sector tendencies, appears to have a large even distribution of incomes for men and women software workers. Portland also has an large even distribution of software workers. However, Portland’s income distributions are at slightly lower income levels and with two distinguishing characteristics for women – larger shares of workers are at the bottom of the income distribution and the top end of the income distribution has an accelerated tapering. Both Austin and Portland have slightly higher income ratios of women to men, indicating a smaller wage gap.

The open-source Marshallian district allows for workers to enter at the bottom of
the labor market. The public-sector tendencies of Austin’s industrial district provide steady promotion opportunities with set wage levels. My public-sector interviewees in Austin were aware that they may get paid less than if they were to seek employment in the private-sector, but they each expressed relief in the stability of the job and the well documented internal hierarchy and career ladder. Of course, not all workers that work in the public sector stay in the public sector, but interviewees expressed that the public-sector provided safe-guards that made the lower pay attractive for women. These incremental wage levels may be the small spurs on the bulge of Austin’s violin plot, evident for men, women, and whites. Unlike the other two regions, the dichotomy between Asian and minority software workers is drastic. Asian software worker incomes concentrate at the high-end of the income distribution and tapers off toward the low-end. The minority income distribution concentrates at the low-end and tapers off in irregular fashion. Austin had a strong community of Hispanic/Latino oriented meetups and groups, and, at 18%, had about twice participation of minorities in the software labor force than Portland or Seattle. Based on conversation – and a topic for future research – I expect the bulge at the bottom to be majority minority women, while the top taper to be majority minority men. While each region had similar income ratios of women to men, Seattle’s being the smallest of the group, the income ratio for minorities to whites was by far the largest. Despite the highest representation of minorities, they appear to be to earning less than minorities in other regions.

Portland and Austin are similar in median wages for software workers. As a whole and across gender, Austin software workers earn $2k–$3k more than their Portland counterparts, but when comparing race/ethnicity, drastic differences emerge. Asian and white software workers in Austin earn about $5k than Portland Asian and white software workers. The median incomes and the violin plot show a reversal for minority software workers: Portland minority software workers make $5k more than Austin minority
Figure 6.4: Income distribution by gender for software workers, Austin, Portland, and Seattle, ACS 2011–2015.

By race/ethnicity, Austin’s software workers have the highest income stratification, ranging between $53k and $90k, whereas Seattle’s ranges between $84k and $96k.

In Portland, minorities and women have the largest share in the lowest portion of the income distribution, yet, once outside this share, there Portland workers exhibit a healthy middle to their income curves. Asian software workers make more than minority counterparts, but again the violin plots are not nearly as drastic as Austin’s. There appears to be a earlier narrowing of income distributions at the high-end of the income distribution. Portland also had the strongest and healthiest on-boarding community, welcoming entrants into software production and giving them guidance. As a interviewee said in the previous chapter, one just needs to show up and someone will be helping them code.

Seattle contrasts to Portland and Austin in that each of its violin plots resemble near
perfect diamonds, with three small notes. First, in stark contrast to Portland and Austin, the diamond shapes of Seattle’s income distributions leave far fewer software workers at the bottom of the income distribution. Second, men, as the highest paid group across any region, had the longest high-end tail in income distributions. Lastly, there appears to be a noticeable flattening at about the $100k mark in the plots for women, Asians, and minorities. The flattening is most noticeable in the plot for women’s income distribution.

On the first note, Seattle’s industrial district is the most mature. Microsoft started at the cusp of, and epitomized, the mass-market software period. The hub-and-spoke nature reflects the labor market practices of Microsoft and now Amazon in the cloud/mobile software period. As a previous interviewer noted, both of firms shape the labor practices, firm strategies, and technologies, of the other, and the spoke-firms in the region. The diamond shapes reflects the mature and steady growth of the two firms, creating managers
and providing clear entry points into the firm. Notably, and unlike Portland and Austin, a software worker would move to Seattle to work for Microsoft or Seattle, rather than moving without a job to learn to code, like one might in Portland or Seattle.

The second and third notes are related. The flattening of incomes, most noticeable for women and likely minority women, is the manifestation of a glass ceiling. Men at Microsoft and Amazon provided the most insight into this phenomenon in interviews. Some viewed it as a matter of fact.

Well we had an internal report about diversity. Our numbers were better than most large tech companies. Most of the women [at large firm] are in non-technical positions ... of course [shrug].

Others explained that because of their interview process, mainly higher-up engineers would do the interviews. Interviewees elaborated by describing similar scenarios. On the day of an candidates interview, a human resources representative, likely a woman, would greet the candidate. After an initial screening, the rest of the day, comprised of interviews conducted by senior employees, likely all white or Asian men. The result was an exacerbation of the gender gap. Only recently have the firms, separately, decided to alter their interview processes.

To be clear, Seattle's numbers in Table 6.4 appear almost as equitable as Portland and Austin in gender, and more equitable in race/ethnicity. As reported in the last chapter, the median Seattle software income is 2.5 times greater than the metro median, higher rate than Portland and Austin. Yet the regional mode of production, mediated through the industrial district, varies the challenges experienced by different groups of workers. Portland and Austin appear the most similar in that the challenges for women are in

---

58Seattle product manager at a large firm. Interview, interview identification withheld.
entering the regional labor market, whereas in Seattle the challenges for women are in moving into top management positions.

In the next section I will dive deeper into these issues by examining sub-labor markets.

6.3.2 Segmentation and dualization of the labor market

To gain finer detail into the labor market, to address the three occupational bifurcations (moments) documented in an earlier section, and to explore feminization of software labor, I group occupations into four tiers of the labor market: management, top, middle, and bottom. I choose this method, instead of a simple “dual” labor market, to examine the long term impacts of the bifurcation of occupations from earlier time periods. A fifth tier collects all other computer occupations which I include for reporting purposes, but it cannot be clearly analyzed without further information about the roles of workers in this occupation. The labor market tiers presented in Table 6.5 are collections of occupation classifications in the last chapter. For example, engineers and specialized analysts comprise the top-tier of the labor market while programmers, analysts, and administrators comprise the middle-tier.

Each tier represents a sub-labor market within software production, having differing educational or experience requirements and rewarded with matching salaries. The management tier is upper management. The top-tier are technical design, architecture, and risk mitigation functions. The middle-tier comprises producing and maintaining technical designs or risk mitigation plans through the development or application of software. The bottom-tier comprises technical work that relies heavily on already produced software.

Similar to the data in the previous chapter, I present the violin plots and summary statistics on gender in each region in Table 6.6 and Figure 6.7 and on race/ethnicity in Table 6.7 and Figure 6.7. Each facet deserves detailed analysis, but the task at hand is to compare the extent of labor market bifurcation and segmentation and variation local
Figure 6.6: Income distribution by labor market tier by gender for software workers, Austin, Portland, and Seattle, ACS 2011–2015.
Figure 6.7: Income distribution by labor market tier by race/ethnicity for software workers, Austin, Portland, and Seattle, ACS 2011–2015.
manifestation. Therefore, I will focus primarily on larger trends. I use interview data to gain further insight into sub-labor markets. I focus primarily on gender in this section, reflecting the topics of choice of my interviewees, even women of color.

Women’s participation in the each tier across regions was fairly similar. Notably, women’s participation in the middle tier was lowest. The middle and bottom tiers also had highest wage gap of the other tiers. Women in Portland’s middle tier had the greatest inequity, but traded that for far higher pay in the top tier. In Austin, women in management had near parity with men.

In Austin and Portland, minority participation taper significantly in the higher tiers. For race/ethnicity, Portland’s bottom tier and top tier were highly unequal, yet the middle and upper management tiers were the best performing among the regions. Seattle had consistent minority representation in each tier, yet, like Austin, had a significant drop in race/ethnicity wage gap in the top tier. Further, Austin had the lowest race/ethnicity wage gap of 0.61:1 in its rather large middle tier.

One takeaway thus far is that while the income distributions looked similar for Austin and Portland, the occupations, and more importantly the sub-labor markets that comprise those income distributions, vary drastically between region. The mechanisms that exist for workers in each region to move up in a tier thus also vary drastically. In terms of purely wage, at no time is moving up a tier a poor choice. Yet, moving up to a higher tier in the labor market is also difficult. Little if no formal training exists from firms at the bottom of the labor market. Instead, workers must rely on organizations and networks outside firm boundaries to climb the career ladder. In places like Seattle, the bottom of the labor market is already small.

In places like Portland and Austin where there is a relatively large bottom tier of the labor market, there is little incentive for firms to train workers. One woman lamented that sexism at the bottom of the labor market expressed this as being expendable to the
firm, and a liability if she was promoted.

They don’t care about us really, like, what’s in it for them [manager]? I already experience harassment in the workplace, they’d rather probably see me out the door [laughter].

Another women took a previous job as a junior developer with the promise of training.

Even though the pay was less, I took the job because they were willing to train me... but then they didn’t train me, any of us. It wasn’t a good environment for women.

Seeking promotion often requires adhering to certain rules of masculinity and confrontation. An earlier quote from a women documented the incentives to propose new solutions to managers in team meetings. She also complained that she didn’t bring up solutions because she didn’t want to have to argue about it with the rest of her team, and in her own words brought up the masculine nature of the workplace, success and, tying to the previous quote, promotion.

I’d rather just stay quiet. I’ll write an email later, I just don’t want to ... argue with the team. It can be hostile and very masculine when you propose something new, especially if they [men on team] don’t know about it already.

Even the non-workplace functions of “tech-culture” present challenges.

I hate the game room. It’s too aggressive and competitive.
And yet, even among women in the workforce, when a women doesn’t express their opinions, or confront sexism, they can receive backlash from other women.

... people like me get it from the other side too – for not addressing sexism eveeeery single time something happens. If I did, it would be the only thing that I do all day, forget working. We should just be able to address the sexist.63

The initial move out of the bottom tier, for women who enter the software labor market at the bottom, is one of the most commonly identified challenge for women software workers. Also common among the women I interviewed was the use of a network to find their current job. Yet to develop that network to get a job outside of the bottom-tier of the labor market was difficult, even though she was well qualified.

It was all friends of friends. And I felt like it was like once you’re in the bloodstream, you’re in, and you can kind of do go anywhere and do anything just by the people you know. [long pause] It took a minute to get into the bloodstream, like breaking in was kind of hard, but once you’re in, Austin is small enough that there’s enough, and there’s enough business that people will quit to go to another company all the time. And now you know someone at that other company.64

Competition still exists at the bottom of the labor market. She, like the other women I interviewed, distinguish themselves from other women who attend coding schools but did not attend college, later commenting that coding schools can only prepare you to do the code, but not the other tacit queues of the labor market.

It doesn’t matter if you went to Yale or UNT [University of North Texas], people are willing to work with you.” Even if you only went to a coding

63 Portland developer at an external SME. Interview, 09/28/2016, D18, transcript. Emphasis original.
64 Austin developer at a large external firm. Interview, 11/11/2015, D1, transcript. Emphasis original.
school ... “people would be willing to work with you, but it’s not the same as someone who went to college where you [pause] learn more than just code.”

Two other women explained it differently.

You can’t have a bootcamp grad [coding school] be your primary developer.

Where you go – how far you go – is about about where you start. Man or woman or whatever. That has certain benefits [chuckle] ... if you’re a college-educated man.

The reconfiguring of education in the US to view software development as a job skill that has potential to create wealth among those with limited access requires scrutiny. During the time of writing this dissertation, quotes from my interview subjects along with two events suggest we rethink this narrative. Without the non-technical training, like integration into a network, navigating promotion, and navigating masculine values, women and minorities entering at the bottom of the labor market are not equipped to climb the career ladder.

Recent work has challenged the idea that for-profit educational institutions, like coding schools, produce workers with adequate skill-sets for the workplace. Instead, for-profit educational institutions prey on marginalized workers seeking to break into lucrative occupations and careers (Cottom 2017). Second, ITT Technical Institute (ITT Tech) closed down its 130 campuses citing trouble with the US and state prosecutors over predatory practices (Moore-Gerety 2016). ITT Tech, founded in 1969, was the oldest technical training institution and well-known as promising skills to workers in technical, and software, fields so that they could get good jobs.

65 Austin developer at a large external firm. Interview, 11/05/2015, D1, transcript.
66 Austin developer at an SME. Interview, 10/03/2015, D2, transcript.
67 Portland developer at an external SME. Interview, 09/28/2016, D18, transcript.
With these insights, the systematic bifurcation of software occupations, and insights from my interviewees, we might conclude that coding schools supply low-skill, low-wage computer work in large volumes. As we saw from Hall, Markusen, et al., we might consider this as assembly line software development – similar to the assembly lines of Detroit. Those workers that enroll in coding schools without college degrees may find themselves trapped in bottom tier occupations, even with the necessary technical skills. Women, and especially women of color, face additional gender and race/ethnicity based racism and sexism.

Yet, even with a single occupation – in a top-tier – the gendering of roles has tangible impacts.

We’re all architects, but they have me doing the work that is customer facing.
I like it better, but it’s all the women doing anything customer facing. I know it’s sexist, but I just kinda gave up on pointing it out every time.68

The interviewee continued and explained that coworkers perceived her job as less skilled than those doing purely technical work, all of whom were men, even though they had the same job titles. She had the same qualifications and she had more experience than at least half of her co-workers. She also complained that the customer facing job was, even though perceived to be less-skilled, was actually a lot harder because it required combination of both technical and social skills, and she exerted significant emotional labor in working with customers.

Three women who interfaced directly with customers also shared stories of interacting with customers who demanded to speak to their boss, one customer even going so far as to ask directly for the man in charge. In one of the cases, the interviewee was highest-level customer-facing employee, and in the another, the interviewee was the head of the

68: Portland developer at an external SME. Interview, 09/28/2016, D18, transcript.
department. To reiterate, they were doing technical software production work alongside customer-facing work.

All three women, spanning each region, mentioned that they were pigeon-holed into the work because of their gender. Further, because their jobs are perceived as less technical, they feel as if they are not rewarded to pursue new skills – addressing the race condition of software workers – like the men in their same roles. This is at heart of how bifurcation negatively impacts women. If the technical portion of their job were to be automated through software, as we expect firms to do, then the men who were able to pursue new skills would have a better chance of moving into the higher-tier occupation and the women who were not able to pursue new skills would become the users of the software. They would be doing technologically based emotional labor.

Referring back to Table B.1, both of the occupations which pay the least require some sort of customer, or client, facing interaction. They also highlight the use of software. These are already perceived as more feminine. Median wages in these occupations for women in both Portland and Austin are less than $35k per year. To move to a middle tier occupation, the median income for women is about $60k in both Portland and Austin. This is where the two regions differ begin to differ. In the middle tier, the median income in Portland is $79k. In Austin it is only $73k for men.

The median income for bottom tier in both regions is in the mid $30ks for minorities. A move to the middle tier in Austin, for minorities, is met with a median income of $46k. A move to the middle tier in, for minorities, Portland is met with a median income of $62k. For Asians the median income in both regions is in the mid $80ks, and for whites the median income is in the mid $70ks.

I end this section on a rather dire note. I showed how the systematic bifurcation of software occupations leads to the race of workers to stay competitive by learning new skills while avoiding being deskilling by other software workers. I showed how sexism
hinders the progress of women within software work, and combined with bifurcation, create gendered occupations and gendered roles within single occupations.

In the next section, I look at local responses, mediated through organizations and institutions, to these forces.

6.4 Local Mediation: Community, Collective Action, and Change

In this section, I discuss the local mediation of labor market dualization and segmentation outcomes. That is, I discuss the local mediation of the structural forces that lead to deskilling, and the gendering and racialization of software work, and the uneven distribution of incomes. Institutions and organizations mediate between individual actions and outcomes and broader society-wide phenomenon. Thus, I focus on local mediation expressed as the practices of local municipalities, firms, labor organization – whether, conferences, events, or communities. I call out specific municipalities, firms, and groups, and use the term labor market intermediaries (LMIs) to refer to the role that they play in within the labor market. I build from my interview data, participant-observation data, and the data presented throughout this dissertation. The type and function of an LMI is dependent on the regional mode of production. Thus, where applicable, I draw on the similarities between regions, like the Marshallian tendencies of Portland and Austin. In other places, I draw on the regionally specific components of the region, like Austin’s public-sector influence, the absence of large firms or a large public-sector, or Seattle’s large hub-firms.

To understand local mediation, I used the transcriptions from my interviews to identify actions that workers take which concretize, challenge, or change the ways that deskilling, segmentation, racism, or sexism impact local labor. I then group these responses by their action and by the mediation. The chapter subsections reflects the type LMI. As such, I draw mostly from my interviews with women and minorities in this chapter and,
following the topics of discussion of my interviewees, I focus primarily on the issues of
deskilling, segmentation, and sexism. Racism did come up, but not in the repeated fashion
that sexism did, nor in references specific to software production. I will address it below.

I close this section by discussing the unique nature of software, stressing the impor-
tance of these types of local mediation for software agglomerations.

6.4.1 Labor- and community-groups
A small number of labor- and community-groups, led and organized by local software
workers, act as important LMIs in each of the regions. Those oriented around specific
software technologies or languages tend to seek combat individual and regional deskilling.
Those oriented around women or minorities combat racism and/or sexism in combination
with deskilling, creating opportunities for women and minorities to climb the career
ladder. They are volunteer run and occasionally receive funding from, but frequently use
space provided by, municipalities and/or firms.

The type and orientation of the labor-led LMIs takes on distinctly local character. For
example, in Austin, one of my interview subjects participated in, or started, the local
PyLadies, PyMoms, AustinFeminist, DjangoGirls while also attending the local Python
Users Group. Another Austin woman had a list that didn’t overlap and included Latina
Hackers, Girl Develop IT, Gem Austin, Girl Start, Women Who Code, Girl Connect,
Texas Girls Collective, Code Dojo, and Girls Who Code, while also attending the local
Ruby Users Group. Other women in Austin attended these meetups in conjunction with
others. These groups reflect the communities active involvement in combating gender
and racialized deskilling, creating networks that allow women to find jobs, and provide
solidarity. Latina Hackers’ main purpose was to combat the intersection of structural
racism and sexism by providing a network of women who could network with each other
– seeking to publicize the group in hopes of “smoking out” other Latina software workers
to build a stronger coalition. The group also engages in providing local Latina teenagers with instruction on software development.

The lists were just as long for some of the women software workers in Portland, but also included women who started entire annual conferences – OSBridge and PyDX – that stress the inclusive nature and selected presenters using metrics of equity as well. As OSCon, mentioned in Chapter 5, moved briefly moved to California for a year, women and minorities from local meetups banded together to create entire annual conferences with equity focuses. OSBridge even cutting across technology boundaries and instead serving the open-source as an ideal. Conference spaces typically hold local Portland conferences, and firm spaces and bars typically host meetups and user groups. Firms like Puppet, Jive, New Relic, and Urban Airship all host meetups as long as an employee is there during the meetup – which sometimes overlaps with the interests of employees.

Some stress the importance of providing women role models to those still in school, like any of the “girl” groups (with the exception of DjangoGirls), but focus on mentoring and providing a way to exchange knowledge within the community. I asked every interview subject how they got started in software and how they grew into their existing career trajectory. Every woman I interviewed brought up mentoring and networking – without me specifically asking. They stressed the importance of mentors and networks in starting or expanding their career. I asked every interview subject how they got started in software and how they grew into their existing career trajectory. Every woman I interviewed brought up mentoring and networking without me specifically asking. While not all LMIs, or even meetups, provide some formal mentoring, having a network of women or persons of color in similar occupations provide an opportunity to share experiences. Most LMIs promote some sort of mentoring and those that have received some mentoring are quick to seek out mentees.

69 Austin developer. Interview, interview identification withheld.
One interviewee directly associated the growth of her career with the insights and network she gained when she found a mentor.

My career finally starting taking off when I found other female product managers mentors.\textsuperscript{70}

Another woman repeatedly mentioned the craft nature of programming, thus necessitating mentors.

Mentors are vital. Programming is a craft.\textsuperscript{71}

Technical skill building act as justification for these groups. It certainly plays an important part. However, interviewees expressed two other important parts of these groups – and LMIs in general: the role of networking, and the role building tacit, non-technical, skills. A woman in Portland pointed out that there are numerous methods of gaining technical skill, but the local network and mentoring are not something you can necessarily reading coding tutorials.

There’s not a lack of learning materials, there’s a lack of mentoring.\textsuperscript{72}

One woman stressed getting over the typical gender role of self-doubt. She learned the importance talking with other women, even informally, about being a woman in a male dominated workplace, placing value on having a network to navigate the workplace.

Get over the self-aggrandizing role and stop asking ‘am I going to embarrass myself?’ That’s a constant topic.\textsuperscript{73}

Another said it differently.

\textsuperscript{70} Austin developer at a large external firm. Interview, 11/05/2015, D1, transcript.
\textsuperscript{71} Austin developer at an SME. Interview, 10/03/2015, D2, transcript.
\textsuperscript{72} Portland developer at an external SME. Interview, 09/28/2016, D18, transcript.
\textsuperscript{73} Portland developer at an SME. Interview, 10/20/2015, D16, transcript.
a space that is comfortable ... to lower your guard a little.\textsuperscript{74} (I need)

Labor-led LMIs like serve important roles. Prior to being successful equity-focused Portland tech conferences, the two groups were informal networks of colleagues that crisscrossed in meetups and user groups. While the interviewees stress the importance of these groups, one woman in the Austin complained that the meeting groups lack the longevity and funding to make permanent impacts. PyDX has yet to approach the Portland Development Commission, but that may be changing soon. The economic development agencies can importantly connect the needs of the local labor market to partners and funding throughout the community, sometimes connecting one group of workers to another group of individuals.

\textbf{6.4.2 Municipality led initiatives}

Each interviewer that mentioned an influential LMI mentioned their local nature. Referring back to the idealized industrial district typology, we recall that workers in hub-and-spoke district workers are loyal to the firm, whereas Marshallian district workers are loyal to district. Similarly, the local-focused, whether firm or district based, was necessary to focus on the immediate needs of the labor market.

Municipalities, and their economic development agencies, play important roles in coordinating firm initiatives, a discussion for the next section, and for liaising between firms and labor groups. This allows for local municipalities fund and give legitimacy to local firm and labor needs. In Austin, Capital Factory does just this, shaping and reflecting the local labor market. It provides spaces for local meetups, provides low-cost startup space, and has a gender and racial diversity requirements as part of its coding school. The original group of Capital Factory founders approached the City of Austin's Economic Development Department for funding of Capital Factory. This was the catalyst to making

\textsuperscript{74}Austin developer at a large external firm. Interview, 11/05/2015, D1, transcript.
the rag-tag group of 17 developers turned local champions – originally only connected by one of the founders – into the mentors for the first class of the startup accelerator, connecting the group to other public and private resources. Today, Capital Factory, as a public-private initiative Local entrepreneurs seek venture capital, local software workers use coding schools to expand from their legacy coding training into new languages like Ruby and python.

In the Seattle region, municipalities primarily work to meet firm needs. In Portland and Austin, municipalities also seek to meet firm needs, but, like Seattle, they focus on large firms which may be technology-oriented, but aren’t necessarily software focused. These are important, but I argue the importance of software-specific LMIs in the last part of this section. In Portland, the successful and local serving OSBridge conference approached the Portland Development Commission for seed funding. They were able to get a small grant and make the first conference a success. PyDX has yet to approach the city, but has found funding from other local firms.

6.4.3 Firms and firm associations

The Seattle region does have community groups, but the community is less organized and overshadowed by the dominance of Amazon and Microsoft. More specifically, the community is less organized across firm boundaries. Instead, both Microsoft and Amazon have internal mentoring programs, internal classes or pay for classes at local universities, and have informal book groups – two mentioned the book written by former Yahoo CEO “Lean In,” which were the best attended book groups that either had participated in, and were happy to report that the group was critical of her work.

To be clear, firms can play important LMI roles that can sometimes mitigate the impacts of labor market segmentation and deskilling. Sometimes though, they can’t – or mitigation is against their best interest. The story of Microsoft’s recent restructuring
to make its employees all developers, phasing out the testing-only engineers, is just one example. Large firms do have both technical groups and groups that are, for example, specifically for women. But large firms, especially more mature ones, still have remnants of a company culture that one might have expected to from IBM or Tektronix. These types of cultures realize the value in investing in labor, within firm boundaries, even if the cost and benefits of which are not easily attributed to a segments profit or loss. These types of programs, of course, require that the company is making enough surplus to reinvest in technology, expansion, and labor. Companies like Amazon, which only recently became consistently profitable, may not have the same types of expectations or culture.

One Austin employee that worked at a branch of a large, mature, external firm said that she goes to all the women-oriented meetings.

There’s actually really good conversation there. And sometimes when we pick our battles, we can, we make change happen like the presentation group.

I ask for funding for [name of local labor-led group].

Workers at every large firm I interviewed, whether headquartered locally or at a branch, reiterated the importance of mentoring. A women in Seattle didn’t care that her mentors were both men, one of them she sought out informally and the other was formally assigned through her on-boarding.

Every place is different. I wanted to know what would be valuable to Amazon. I would not have been able to move from front-end to my position without their help.

---

75 Austin developer at a large external firm. Interview, 11/05/2015, D1, transcript.
76 Seattle developer at a large firm. Interview, 10/27/2016, D27, transcript.
Another woman in Austin sought the mentoring of a woman at a local branch of a large external software oriented firm. She related her happiness in her current job to her learning how to handle various situations from her mentor.

It’s simple. If you’re not happy, you’re not going to stay.\textsuperscript{77}

And often it is in the firms best interest to support mentoring, skill-building, and networking, even if it crosses firm boundaries. It often benefits firms in ways that manager may not be able to identify, hence the importance, as one interviewee put it to stress the technical skills over tacit skills when seeking firm support. The hard technical skills are what the managers of the software workers I interviewed think are important to their job.

Firms also play important actors in creating or participating in multi-firm led initiatives. In Portland, led by Puppet, a group of the largest (small) software firms alongside branch plants of other software firms, banded together to create a diversity pledge. The pledge is to improve diversity in women and minorities in technical positions. One of my interview subjects advised on the measures of success and the method of reporting. One of the key issues was salary transparency – a sticking point for some of the firms – which was later made optional.

\textbf{6.4.4 What makes this unique to software?}

LMIs are, of course, important across any laboring activity or industry. Two items make LMIs more important for software. The first is that software production produces more software. The second is the already low inclusion rates of women and minorities. These two characteristics feed into each other.

First, as I’ve shown at length, software production is unique because software produces software. Software is a tool of automation used to increase efficiency. Its low-materiality

\textsuperscript{77}Austin developer at an SME. Interview, 10/03/2015, D2, transcript.
means that its production requires little material input and that drastically reconfiguring and replacing its purpose or function can be a few lines of code. Thus the speed of software development is rapid compared to other types of production because human knowledge, through living labor and the dead labor stored in existing software, is the primary input. With the internet, the distribution can be instantaneous and free. There is no need to ship a design to manufacturer to produce prototypes, send them back for approval, and so on until it gets into a store where a customer visits to purchase the physical commodity. The low-materiality means that firms’ costs are overwhelmingly labor related. Software workers, which have only their labor to sell, produce new software. Firms and workers use software to improve the efficiency of software production (recall the back-end developer asking if he could just use a front-end library instead of having to work with the front-end developers). This means that when software workers make software for firms, they may be deskilling themselves, their colleagues, software workers in another country, or society in general. Consequently, software workers are under constant pressure to reskill themselves by learning new software libraries or languages.

Second, the already dismal inclusion rates of minorities and women are unlikely to improve on their own without significant structural changes – if anything, they seem to reinforce barriers for marginalized groups seeking entrance into the software labor market. Certainly feminization offers one avenue of change, but because of the first issue, women and minorities are at a disadvantage in skillbuilding that will meaningfully impact inclusion or wages. The metaphorical rungs on the ladder that lead to climbing into the top-tier of the labor market are few and far between. Instead, women and minorities will continue to concentrate at the bottom of the labor market. Certainly there are broader social issues at play, namely blatant bigotry, but software production is already infamously dominated by white men.

These items place increased importance on the role of collective action. Collective
action, as a labor group, through government, or at the firm, allows for software workers to decide how and what software to produce – enabling software workers to engage in the question “software production for whom?” In this research, the strongest collective action groups were the groups that sought to address issues of inclusion first, then technical skillbuilding second. Building stronger local labor-led groups helps to mitigate the negative structural tendencies toward rapid deskilling and the racialization/gendering of software work. Those groups which seek to address inclusion are most likely to not only mitigate negative impacts of labor market dualization and segmentation, but also to create software for the district, a scenario that happened in Portland in the late-2000s and early-2010s. These are what led to Calagator, a calendar to track local events, and WriteTheDocs, a set of local conferences to improve documentation of open-source software – both tools vital for new entrants to Portland’s labor market and to software in general.

I’ll end this section with a quote from a women from Austin about why she participates in so many different groups.

What kind of culture do you want to make? [You] have to create these moments.  

6.5 Conclusion

In this chapter, I sought to understand the labor market dualization and segmentation of software production, its local manifestations, and how it is locally mediated through institutions and practices. Specifically, I asked two questions:

1. To what extent has software production demonstrated tendencies toward labor market dualization and segmentation by gender and race/ethnicity?

2. How are these labor market outcomes mediated by local institutions and practices?

78 Austin developer at an SME. Interview, 10/03/2015, D2, transcript.
To address the first question, I iterated on the IPUMS dataset, adding race and gender as key variables to the growing list of income, occupation, and metro region and again analyzed interview and participant-observation data. I present the historical trends of bifurcation in the software occupations by developing Tables 6.1 and Figure 6.1. I then combined occupations into like groups, forming labor market tiers. I use these four tiers to understand gender and race/ethnicity dynamics, addressing the feminization of the labor market. I showed how five moments of occupational bifurcation manifest as labor market dualization. I then showed how these structural forces are mediated through the regional mode of production. I show that the low-skill software labor market is distinctly more female and more non-white and that there are fewer opportunities for workers in the bottom tier of the labor market to advance.

Lastly, to address the last question, I found similarities in the experiences of software workers and outlined how local responses, through labor-groups, local government, firms, or combinations of these, give labor greater power to mitigate negative impacts that I identified addressing the first question.

Building on this research, I’ll repeat the claims I made at the beginning of the chapter.

1. The product of software labor is software, and software is the same “machine” used to automate software work. The rapid pace of software development thus coincides with the rapid deskilling of software workers. Software workers thus implicate themselves in creating programs which deskill themselves and other software workers.

2. Regionally specific modes of production mold the dualization and segmentation of the local software labor market. Nationally, the share and number of women and persons of marginalized racial/ethnic backgrounds is growing, but primarily at the lowest tier of the labor market. Occupations in the lowest tier are more precarious
than their higher-skilled counterparts, require less skill and provide less income. Local variations on national trends depend on the regional mode of production.

3. Labor market intermediaries — through initiatives and groups within large firms like the example in Seattle, through community-led events like the example in Portland, or through community-organizations like the example in Austin — play a important roles in enabling marginalized workers to build technical skills and gain soft, or tacit, skills to better navigate the labor market.

This chapter argues that regional divergence in software production shape the form of, and the responses to, labor market dualization and segmentation. I showed that labor market dualization and segmentation manifest differently in each region. I showed that local responses, mediated by local institutions and local practices, are necessary to mitigate the detrimental impacts of the local manifestations of segmentation and bifurcation. Ultimately, however, this chapter argues that collective action is a necessary response to mitigate broader structural forces that detrimentally impact women and minorities.
Table 6.3: Median income by race/ethnicity for software workers, 1970-2015.


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of workforce</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asian, NH</td>
<td>5,200</td>
<td>21,820</td>
<td>89,711</td>
<td>386,098</td>
<td>558,665</td>
<td>769,133</td>
</tr>
<tr>
<td>White, NH</td>
<td>273,600</td>
<td>476,500</td>
<td>1,081,902</td>
<td>2,493,945</td>
<td>2,574,136</td>
<td>2,814,476</td>
</tr>
<tr>
<td>Minority, all other</td>
<td>12,200</td>
<td>41,640</td>
<td>117,371</td>
<td>414,354</td>
<td>485,494</td>
<td>661,362</td>
</tr>
<tr>
<td>Percent minority</td>
<td>4%</td>
<td>8%</td>
<td>9%</td>
<td>13%</td>
<td>13%</td>
<td>16%</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asian, NH</td>
<td>62,178</td>
<td>59,886</td>
<td>68,174</td>
<td>79,710</td>
<td>84,156</td>
<td>85,000</td>
</tr>
<tr>
<td>White, NH</td>
<td>66,126</td>
<td>61,549</td>
<td>65,877</td>
<td>72,464</td>
<td>74,220</td>
<td>72,262</td>
</tr>
<tr>
<td>Minority, all other</td>
<td>52,967</td>
<td>53,234</td>
<td>58,435</td>
<td>60,870</td>
<td>61,935</td>
<td>60,000</td>
</tr>
<tr>
<td>Income ratio</td>
<td>0.80</td>
<td>0.86</td>
<td>0.89</td>
<td>0.84</td>
<td>0.83</td>
<td>0.83</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All software workers</td>
<td>66,126</td>
<td>166,304</td>
<td>35,460</td>
<td>48,116</td>
<td>74,220</td>
<td>72,262</td>
</tr>
</tbody>
</table>

(Table 6.3, Figure 6.3 and Figure 6.2) Minority excludes Asian non-Hispanic, listed separately. Incomes in 2015 USD. Author’s calculations from data gathered from IPUMS-USA (Ruggles, Genadek, et al. 2015) ACS 2011–2015. Software occupations include OCC codes 110, 1005, 1006, 1007, 1010, 1020, 1030, 1050, 1060, 1105, 1106, 1107, 1200, 1220.
Table 6.4: Median income by gender and race/ethnicity for software workers, Austin, Portland, Seattle, ACS 2011–2015

<table>
<thead>
<tr>
<th>Gender and race/ethnicity dynamics for software workers.</th>
<th>Austin</th>
<th>Portland</th>
<th>Seattle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of workforce</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Men</td>
<td>43,511</td>
<td>33,287</td>
<td>93,881</td>
</tr>
<tr>
<td>Women</td>
<td>11,437</td>
<td>10,083</td>
<td>26,055</td>
</tr>
<tr>
<td>Percent women</td>
<td>21%</td>
<td>23%</td>
<td>22%</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Men</td>
<td>77,424</td>
<td>75,297</td>
<td>98,070</td>
</tr>
<tr>
<td>Women</td>
<td>66,140</td>
<td>63,232</td>
<td>80,092</td>
</tr>
<tr>
<td>Income ratio</td>
<td>0.85</td>
<td>0.84</td>
<td>0.82</td>
</tr>
<tr>
<td><strong>Size of workforce</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asian, NH</td>
<td>9,129</td>
<td>5,764</td>
<td>31,680</td>
</tr>
<tr>
<td>White, NH</td>
<td>35,965</td>
<td>33,074</td>
<td>78,211</td>
</tr>
<tr>
<td>Minority, all other</td>
<td>9,854</td>
<td>4,532</td>
<td>10,045</td>
</tr>
<tr>
<td>Percent minority</td>
<td>18%</td>
<td>10%</td>
<td>8%</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asian, NH</td>
<td>90,104</td>
<td>85,000</td>
<td>96,666</td>
</tr>
<tr>
<td>White, NH</td>
<td>75,359</td>
<td>70,081</td>
<td>94,631</td>
</tr>
<tr>
<td>Minority, all other</td>
<td>52,694</td>
<td>57,963</td>
<td>84,310</td>
</tr>
<tr>
<td>Income ratio</td>
<td>0.70</td>
<td>0.83</td>
<td>0.89</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All software workers</td>
<td>75,359</td>
<td>72,000</td>
<td>94,000</td>
</tr>
</tbody>
</table>

(Figure 6.4, Figure 6.5, and Table 6.4) Minority excludes Asian non-Hispanic, listed separately. Incomes in 2015 USD. Author’s calculations from data gathered from IPUMS-USA (Ruggles, Genadek, et al. 2015) ACS 2011–2015. Software occupations include OCC codes 110, 1005, 1006, 1007, 1010, 1020, 1030, 1050, 1060, 1105, 1106, 1107, 1200, 1220.)
<table>
<thead>
<tr>
<th>Codes</th>
<th>Titles</th>
<th>Classification</th>
<th>Median Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>Computer &amp; information systems managers</td>
<td>Managers</td>
<td>95,000</td>
</tr>
<tr>
<td>Top-Tier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1020</td>
<td>Software developers</td>
<td>Engineers</td>
<td>90,000</td>
</tr>
<tr>
<td>1106</td>
<td>Network architect</td>
<td>Engineers</td>
<td>90,000</td>
</tr>
<tr>
<td>1200</td>
<td>Actuaries</td>
<td>Specialized Analysts</td>
<td>96,005</td>
</tr>
<tr>
<td>1007</td>
<td>Information security analyst</td>
<td>Specialized Analysts</td>
<td>80,092</td>
</tr>
<tr>
<td>Middle-Tier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>Computer research scientist</td>
<td>Programmers</td>
<td>74,085</td>
</tr>
<tr>
<td>1010</td>
<td>Computer programmers</td>
<td>Programmers</td>
<td>73,771</td>
</tr>
<tr>
<td>1006</td>
<td>Computer scientist &amp; systems analyst</td>
<td>Analysts</td>
<td>71,000</td>
</tr>
<tr>
<td>1220</td>
<td>Operations research analyst</td>
<td>Analysts</td>
<td>71,320</td>
</tr>
<tr>
<td>1060</td>
<td>Database administrators</td>
<td>Administrators</td>
<td>71,227</td>
</tr>
<tr>
<td>1105</td>
<td>Network &amp; computer systems administrators</td>
<td>Administrators</td>
<td>64,074</td>
</tr>
<tr>
<td>Bottom-Tier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td>Web developers</td>
<td>Coders</td>
<td>42,155</td>
</tr>
<tr>
<td>1050</td>
<td>Computer support specialist</td>
<td>Coders</td>
<td>46,454</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1107</td>
<td>All other computer occupations</td>
<td>Other</td>
<td>60,000</td>
</tr>
</tbody>
</table>

Note: See Table B.1 in Appendix B for descriptions, incomes, and more information.

Table 6.6: Median income labor market tiers by gender for software workers, Austin, Portland, Seattle, ACS 2011–2015

<table>
<thead>
<tr>
<th>Tier</th>
<th>Gender</th>
<th>Austin</th>
<th>Portland</th>
<th>Seattle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of workforce</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>Men</td>
<td>4,784</td>
<td>3,808</td>
<td>8,948</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>1,881</td>
<td>1,597</td>
<td>3,920</td>
</tr>
<tr>
<td></td>
<td>Percent women</td>
<td>28%</td>
<td>30%</td>
<td>30%</td>
</tr>
<tr>
<td>Top</td>
<td>Men</td>
<td>16,677</td>
<td>11,819</td>
<td>49,322</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>3,322</td>
<td>2,752</td>
<td>9,538</td>
</tr>
<tr>
<td></td>
<td>Percent women</td>
<td>17%</td>
<td>19%</td>
<td>16%</td>
</tr>
<tr>
<td>Middle</td>
<td>Men</td>
<td>11,380</td>
<td>8,554</td>
<td>19,743</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>3,291</td>
<td>2,937</td>
<td>7,512</td>
</tr>
<tr>
<td></td>
<td>Percent women</td>
<td>22%</td>
<td>26%</td>
<td>28%</td>
</tr>
<tr>
<td>Bottom</td>
<td>Men</td>
<td>7,877</td>
<td>7,107</td>
<td>10,980</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>2,117</td>
<td>2,043</td>
<td>3,678</td>
</tr>
<tr>
<td></td>
<td>Percent women</td>
<td>21%</td>
<td>22%</td>
<td>25%</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>Men</td>
<td>90,104</td>
<td>96,110</td>
<td>110,000</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>89,543</td>
<td>82,094</td>
<td>90,104</td>
</tr>
<tr>
<td></td>
<td>Income ratio (women:men)</td>
<td>0.99</td>
<td>0.85</td>
<td>0.82</td>
</tr>
<tr>
<td>Top</td>
<td>Men</td>
<td>95,000</td>
<td>90,104</td>
<td>108,393</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>80,092</td>
<td>81,148</td>
<td>94,849</td>
</tr>
<tr>
<td></td>
<td>Income ratio (women:men)</td>
<td>0.84</td>
<td>0.90</td>
<td>0.88</td>
</tr>
<tr>
<td>Middle</td>
<td>Men</td>
<td>73,084</td>
<td>79,041</td>
<td>90,560</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>61,939</td>
<td>59,068</td>
<td>72,717</td>
</tr>
<tr>
<td></td>
<td>Income ratio (women:men)</td>
<td>0.85</td>
<td>0.75</td>
<td>0.80</td>
</tr>
<tr>
<td>Bottom</td>
<td>Men</td>
<td>42,000</td>
<td>42,155</td>
<td>55,063</td>
</tr>
<tr>
<td></td>
<td>Women</td>
<td>34,778</td>
<td>34,596</td>
<td>46,807</td>
</tr>
<tr>
<td></td>
<td>Income ratio (women:men)</td>
<td>0.83</td>
<td>0.82</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>All software workers</td>
<td>75,359</td>
<td>72,000</td>
<td>94,000</td>
</tr>
</tbody>
</table>

(Table 6.6 and Figure 6.7). Incomes in 2015 USD. Author’s calculations from data gathered from IPUMS-USA (Ruggles, Genadek, et al. 2015) ACS 2011–2015. Software occupations include OCC codes 110, 1005, 1006, 1007, 1010, 1020, 1030, 1050, 1060, 1105, 1106, 1107, 1200, 1220.
## Table 6.7: Median income labor market tiers by race/ethnicity for software workers, Austin, Portland, Seattle, ACS 2011–2015

<table>
<thead>
<tr>
<th>Tier</th>
<th>Race/ethnicity</th>
<th>Austin</th>
<th>Portland</th>
<th>Seattle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of workforce</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>Asian, NH</td>
<td>810</td>
<td>332</td>
<td>2,306</td>
</tr>
<tr>
<td>Management</td>
<td>White, NH</td>
<td>4,906</td>
<td>4,781</td>
<td>9,523</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>949</td>
<td>292</td>
<td>1,039</td>
</tr>
<tr>
<td></td>
<td>Percent minority</td>
<td>14%</td>
<td>5%</td>
<td>8%</td>
</tr>
<tr>
<td>Top</td>
<td>Asian, NH</td>
<td>4,959</td>
<td>3,303</td>
<td>20,877</td>
</tr>
<tr>
<td></td>
<td>White, NH</td>
<td>12,106</td>
<td>9,824</td>
<td>33,619</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>2,934</td>
<td>1,444</td>
<td>4,364</td>
</tr>
<tr>
<td></td>
<td>Percent minority</td>
<td>15%</td>
<td>10%</td>
<td>7%</td>
</tr>
<tr>
<td>Middle</td>
<td>Asian, NH</td>
<td>2,018</td>
<td>1,186</td>
<td>5,276</td>
</tr>
<tr>
<td></td>
<td>White, NH</td>
<td>9,980</td>
<td>9,137</td>
<td>19,726</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>2,673</td>
<td>1,168</td>
<td>2,253</td>
</tr>
<tr>
<td></td>
<td>Percent minority</td>
<td>18%</td>
<td>10%</td>
<td>8%</td>
</tr>
<tr>
<td>Bottom</td>
<td>Asian, NH</td>
<td>771</td>
<td>604</td>
<td>2,281</td>
</tr>
<tr>
<td></td>
<td>White, NH</td>
<td>6,861</td>
<td>7,063</td>
<td>11,001</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>2,362</td>
<td>1483</td>
<td>1376</td>
</tr>
<tr>
<td></td>
<td>Percent minority</td>
<td>24%</td>
<td>16%</td>
<td>9%</td>
</tr>
<tr>
<td><strong>Median income</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>Asian, NH</td>
<td>103,232</td>
<td>130,150</td>
<td>100,000</td>
</tr>
<tr>
<td>Management</td>
<td>White, NH</td>
<td>90,000</td>
<td>88,000</td>
<td>106,000</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>67,077</td>
<td>81,093</td>
<td>84,310</td>
</tr>
<tr>
<td></td>
<td>Income ratio (minority:white)</td>
<td>0.75</td>
<td>0.92</td>
<td>0.80</td>
</tr>
<tr>
<td>Top</td>
<td>Asian, NH</td>
<td>92,106</td>
<td>94,849</td>
<td>101,172</td>
</tr>
<tr>
<td></td>
<td>White, NH</td>
<td>94,849</td>
<td>87,747</td>
<td>110,127</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>87,508</td>
<td>68,502</td>
<td>100,135</td>
</tr>
<tr>
<td></td>
<td>Income ratio (minority:white)</td>
<td>0.92</td>
<td>0.78</td>
<td>0.91</td>
</tr>
<tr>
<td>Middle</td>
<td>Asian, NH</td>
<td>87,000</td>
<td>85,000</td>
<td>87,472</td>
</tr>
<tr>
<td></td>
<td>White, NH</td>
<td>75,000</td>
<td>71,000</td>
<td>84,310</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>46,000</td>
<td>61,939</td>
<td>72,262</td>
</tr>
<tr>
<td></td>
<td>Income ratio (minority:white)</td>
<td>0.61</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>Bottom</td>
<td>Asian, NH</td>
<td>66,140</td>
<td>64,004</td>
<td>55,745</td>
</tr>
<tr>
<td></td>
<td>White, NH</td>
<td>40,000</td>
<td>40,701</td>
<td>52,648</td>
</tr>
<tr>
<td></td>
<td>Minority, all other</td>
<td>38,444</td>
<td>34,778</td>
<td>45,000</td>
</tr>
<tr>
<td></td>
<td>Income ratio (minority:white)</td>
<td>0.96</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Median income

All All software workers 75,359 72,000 94,000

(Table 6.7 and Figure 6.7). Minority excludes Asian non-Hispanic, listed separately. Incomes in 2015 USD. Author’s calculations from data gathered from IPUMS-USA (Ruggles, Genadek, et al. 2015) ACS 2011–2015. Software occupations include OCC codes 110, 1005, 1006, 1007, 1010, 1020, 1030, 1050, 1060, 1105, 1106, 1107, 1200, 1220.  

232
Figure 6.8: Count of software workers in each labor market tier, Austin, Portland, and Seattle, ACS 2011–2015.
7 Conclusion

This mixed-methods dissertation sought to understand the dynamics of software production in regional economies. To do this, I embarked on a journey to trace and analyze the changing nature of software production, the changing geography of software production, the regionally-specific modes of production, and the tendencies to contribute to gendered and racialized inequity. In the end, I found that software workers are hypersensitive to capitalist logic because of the unique nature of software and software production. Workers seek network and skilling-based communities to avoid risk exposure. The place-based nature of software production and these communities provide avenues to combat deskilling, and dualization and segmentation along lines of race/ethnicity and gender. I’ll review some of the key findings before returning to a brief discussion of the need of collective action in software work.

7.1 Summary of Findings

I argued that software production is like management or clerical work. Therefore, software is best measured as a laboring activity that is integral in both new and established production processes. Software labor is integral in industries ranging from health care, food production, biotechnology, finance, education, and extractive industries. I showed that software workers were 0.35% in 1970 and that the 4.3 million software workers in 2011–2015 comprise 2.66% of the labor force. I introduced a fourth period of software
production that draws on the connected nature of personal devices, web/mobile apps, and cloud computing. Each of the software periods represent the products cycles of firms that heavily rely on software during that time, some of which are able to reconfigure themselves across periods. Management persistently pressures software and software workers increase efficiency in order reduce labor costs. Changes in the workplace and software production itself transform the capital-labor relationship in software production from machinofacture to post-Foridst, with evidence today of a computational flexible relationship.

In each software period, new regional software agglomerations emerge, reordering the relationships between regions. Regional modes of software production – locally specific software production – became more divergent over time. Specifically, in 1970 a region’s level software production was dependent on mostly on the size of the region. By, 2015 regions had thriving and differentiated software industrial districts. As firms and workers in each of these second-tier (not Silicon Valley) regions develop new software, they diverge across production (technologies, skill-sets, labor markets), social relations (class-based firm-worker relationships), and/or spatial organizations (industrial district formations). Different modes of production, situated in a regions historical trajectories produce different sets of software occupations and income distributions. I examined the history of my three case regions, their industrial districts, and the ways in which their software firms and workers produce software. Specifically, I analyzed each region’s forces of production (technologies, skill-sets, labor markets), social relations (class-based firm-worker relationships), and spatial organizations (industrial district formations). With this data, I created industrial district types to reflect my three cases regions and detailed their industrial composition. Seattle exhibits tendencies of a hub-and-spoke industrial district, Portland exhibits tendencies of a Marshallian industrial district, and Austin exhibits tendencies of a mixed industrial district. For each region, I demonstrated how historical
developments resulted in software industrial districts with regionally specific modes of production.

Second-tier cities challenge the dominance of Silicon Valley’s software agglomeration. The fates of regional software agglomerations lies in the ability of the firms and workers in the industrial districts to adapt to the new profit cycles of new software periods. For example, Microsoft’s dominance in the mass-market software period did not ensure success in the cloud-and-mobile software period, but with new leadership, Microsoft is starting to adapt. Amazon emerged as a cloud-and-mobile software firm and may have helped build regional specialization in cloud and mobile technologies. Similarly, Portland and Austin had small software agglomerations that emerged from different historical trajectories in the 2000s. Both regions benefited from adjacent industries, and Portland benefited from the development of an open-source network of software workers across industries.

Software production has historically undergone bifurcation, resulting in labor market dualization and segmentation. Five moments of occupational bifurcation manifest as labor market dualization. Four tiers of the software labor market can be to understand gender and race/ethnicity dynamics, addressing the feminization of the labor market. Structural forces are mediated through the regional mode of production. I show that the low-skill software labor market is distinctly more female and more non-white and that there are fewer opportunities for workers in the bottom tier of the labor market to advance. Women and minorities are systematically disadvantaged in the software labor market as bifurcation privileges those at the top of the labor market. With already low inclusion rates, women and minorities find themselves stuck at the bottom of the labor market. Yet, there are similarities in the experiences of software workers that develop local responses, through labor-groups, local government, firms, or combinations of these, to give labor greater power to mitigate these negative impacts.
The product of software labor is software, and software is the same “machine” used to automate software work. The rapid pace of software development thus coincides with the rapid deskilling of software workers. Software workers implicate themselves in creating programs which deskill themselves and other software workers. To fight these tendencies, collective action is necessary. Collective action allows for software workers to decide how and what software to produce – enabling software workers to engage in the question “software production for whom?” In this research, the strongest collective action groups were the groups that sought to address issues of inclusion first, then technical skillbuilding second. Building stronger local labor-led groups helps to mitigate the negative structural tendencies toward rapid deskilling and the racialization/gendering of software work. Those groups which seek to address inclusion are most likely to not only mitigate negative impacts of labor market dualization and segmentation, but also to create software for the district.

7.2 Collective Action

I opened the dissertation with a quote, perhaps a plea, to learn how to code from President Barack Obama. But without collective action – like the important roles that labor-led groups in the last chapter provide to enable marginalized workers to build technical skills and gain soft/tacit skills to better navigate the labor market – regional software work resorts to reducing inclusion of women and minorities. I showed that regionally specific modes of production mold the dualization and segmentation of the local software labor market. Nationally, the growth of women and persons of marginalized racial/ethnic backgrounds in software occupations is encouraging, but growth is “quarantined” at the lowest tier of the labor market.

Occupations in the lowest tier are more precarious than their higher-skilled counterparts, require less skill and provide less income. Local variations on national trends depend
on the regional mode of production. Even still, because of the structure of the labor market and the rapid pace of software development, the benefits of software production do not reach across all software workers. I use the stories in this dissertation to describe the mechanisms, structural and social, that result in this inequity. In chapter 6, I show that the median incomes of women and minorities in the bottom-tier of the software labor market is about $35k a year – less than the amount required for self-sufficiency in each of my case study regions and in stark contrast to the idea that software work produces wealth.

Noble (1986), Braverman (1998), Massey (1995) research other technology-oriented industries. Auto workers, for example, were once highly admired and performed highly skilled work. As the auto industry matured and automation became more pervasive, a similar bifurcation happened amongst automakers. We now use the auto worker as the classic “retrain the autoworker to program” when we talk about tech. In this sense, these structural forces aren’t new. What is new is the rate at which they feedback into the labor market. I noted earlier that this same phenomenon is what allows software profit cycles to advance so quickly, and why software production, as it continues to evolve, provides glimpses into the ways that capitalism in general may evolve and how it might affect other forms of production. This should sound an alarm. Even if we retrained autoworkers, their positions will likely be just as precarious, if not more so, than when they were autoworkers. With just a few lines of code, we could negate the need for them altogether.

7.3 Interventions

The state and labor can serve important roles to intervene, mitigate, and potentially reverse some of the structural forces outlined in this dissertation. Examples from each region present preliminary blueprints for such interventions. For example, the highly networked laborers in Portland used their power across firms and across industry to
learn from each other, elevate the status of the region, and pressure local firms to pledge
to increase diversity and diversity transparency. Local economic development agencies
promote these numbers, further attracting and growing the number and career trajectories
of traditionally marginalized software workers. In regions with hub firms, like Seattle,
workers use their power within the firm to raise awareness around gender through book
groups. In Austin, networked workers seek to ensure that traditionally marginalize groups
have the same access to technical education and career networking that their counterparts
do. The public-private partnership model in Austin provides a blueprint for both Seattle
and Portland to continue to support local entrepreneurs.

While these examples are only blueprints, other more generalizable observations
emerged from this research for economic development agencies. First, connecting workers
across firm, industry, and public-private divides are vital to increase the power of laborers.
At first, this might seem counterproductive to firms, and capital, but stronger labor
networks only increase the “sticky-ness” of regional software production, fueling further
agglomeration. Second, the role economic development agencies have in promoting
locally-serving software conferences that focus on inclusivity and regionally-specific
technologies have direct impact for both laborers, firms, and the interconnected nature of
both laborers and firms. The benefits may be obvious in regions like Austin and Portland,
but even in Seattle, these events have the potential to facilitate and improve relationships
between large firms, small firms, and individual contractors – potentially improving equity
and helping create local and specialized knowledge. In Portland, economic development
support came in the form of money to sponsor local events, but support might come
in seeking to create the networks necessary to develop events altogether. Finally, local
economic development agencies have responsibilities to pressure software firms, large and
small, to report numbers of equity and diversity in technical software roles. These could
be tied to easements or tax credits.
Each of these intervention points seeks to increased labor power and growth local software agglomerations. Increased labor power might help workers combat equity and raise awareness about the deskilling nature of software work in general. Increased inclusivity is a key ingredient in “innovation” – but also in creating better social outcomes.

7.4 Future Research

There are several avenues for future research as a result of this work. The most immediate would be to extend the intensive research to new regions. Of particular interest is the regions of DC and Baltimore because of their similarities to Seattle and Portland respectively. DC is public-sector dominated and Baltimore has a smaller startup scene similar to, but different than, Portland or Austin. Importantly, equity and inclusion are of key concern in both regions. Second, the addition of non-PUMS data sources might provide additional insights into the dynamics observed in this dissertation and might include data from venture capital totals from Thomson Reuters, regional top firms by industry from RefUSA, firm/establishment size from County Business Patterns, industry employment in HT/IT industries, or detail from the National Establishment Time-Series data. Third, expanding on the capitalist variants seems to be larger and longer, but important, component of this research. Finally, examining the specific role of masculinity and the internalization of the neoliberal subject withing software production provide an altogether new area for research.


Cortright, Joseph and Heike Mayer (2002). “Signs of Life—the Growth of Biotechnology Centers in the US, the Brookings Institution Center on Urban and Metropolitan Policy”. In: Washington, DC.


Sayer, Andrew and Richard Walker (1992). “The New Social Economy: Reworking the Division of Labor”. In:


‘Critical GIS,’ in the form of intricately interwoven affinities advocated above, can help us constructively engage not only mainstream GIScience and the ever-proliferating intersections of computation with space and place but also critical human geography.

Thatcher et al. 2016, 821

A.1 Introduction

This appendix seeks to document the a method that assigns a portion of a PUMA geography’s population to its overlapping metro area according to the US Census Bureau’s 2013 definitions. For each 10-year Census period and PUMA geography period, the proportion of PUMA’s population living within the metro area determines the portion of a PUMA’s population assigned to its corresponding metro area. In this calculation, the US Census census-tract-level population for each period (for example, 1970 census-tract populations for the 1970 dataset, and the 2010 census-tract population for the ACS 2005-2009 and ACS 2011–2015) determines the distribution of a PUMA’s population. The result is a crosswalk for the IPUMS PUMA-based data and corresponding US metropolitan areas for four US Census time periods – 1970, 1980, 1990, and 2000 – and two recent American Community Survey samples – 2005-2009 and 2011–2015. The research in this chapter focuses on the 51 US metros with 2010 US Census populations of over 1 million residents (large metros).
This appendix will also detail the steps taken to perform this. Longitudinal Tract Database (LTDB) which harmonizes tract boundaries for 1970, 1980, 1990, and 2000 into the 2010 tract boundaries. The LTDB was chosen, rather than using the original tract boundaries, to further assist scholars that are performing historical tract-level research in combination with metropolitan-level research.

A.2 Setup and Data Sources

A.2.1 Longitudinal Tract Database

Scholars from Brown University, University of Wisconsin at Milwaukee, and Florida State University publish the LTDB (see Logan, Xu, and Stults 2014; Logan, Stults, and Xu 2016). The database provides multiple functions, including the capacity to create tract-level data in 2010 boundaries with any dataset that is in 1970 or later. New functionality allows researchers to convert 2010 data into 2000 level data. For the exercise at hand, we seek to use population data using 2010 tract-level boundaries for the years 1970, 1980, 1990, 2000, and 2010. I downloaded LTDB population data and combined each year into a single shapefile before importing. In this code, ltdb_albers contains the actual tract boundaries and ltdbxy_albers contains the tracts converted to point features (performed prior to this exercise).

A.2.2 PUMA Geographies

These are the geographies used by IPUMS USA. Note that in 1970 and 1980 these are called County Groups, yet are commonly referred to as PUMAs. Each of these files were downloaded from the IPUMS USA GIS Boundary Files web repository.

- 1980 5% County Group boundary file in puma1980.
A.2.3 Metropolitan Statistical Areas

Metropolitan shapefiles (also CBSA) were downloaded from the US Census Bureau website. This exercise used the 2015 definition at the 1:500,000 resolution, using the file `cb_2015_us_cbsa_500k.zip`. The Micropolitan Statistical Areas were removed and only the Metropolitan Statistical Areas were used by filtering LSAD for values of M1.

A.3 Overview

The desired output of the work in this appendix is a table which provides a weighted crosswalk between metropolitan area and each PUMA (or county group) which intersects that metropolitan area. Five tables are generated for 2015 metropolitan area boundaries for county groups in 1970, county groups in 1980, PUMAs 1990, PUMAs in 2000, and PUMAs in 2010. The `popshare` column contains a number between greater than 0 and a maximum of 1. A value of 1 means that the entire population of the PUMA is contained in the metropolitan area. A value of .5 means half the population of the PUMA is contained in the metropolitan area. There are no 0 values as those are excluded. The `popshare` value is calculated using population weighting and is based on the population of the census tracts in that PUMA and MSA in the census year.

An alternative method for calculating this share would be an area weighted share. An example, using the 2015 boundaries for Orlando-Kissimmee-Sanford, FL Metropolitan Statistical Area and the 1970 county groups, might help illustrate how this might work. See Figure A.1. The metropolitan area engulfs (shown in black) engulfs County Group (CG) 3201, but only partially takes up CG 3202 and CG 3203. How might we apportion
the population of CG to the metro area? An area weighted method would look at the fact that the metro area that is in CG 3203 is 3,899 square kilometers (sqkm). CG 3202 is 6,715 sqkm. We would then apply a ratio of CG 3202 toward the metro area of 3,899 sqkm / 6,715 sqkm, or 0.58. For the IPUMS USA data, then, we would scale the person weight (PERWT) of CG 3202 for CBSA 36740 (Orlando Metropolitan Statistical Area) to 0.58. This introduces a couple of problems. First, we assume that the population of CG 3203 is distributed homogenously across the country group. Second, we assume that the population characteristics are distributed evenly across the population. Let us tackle the first issue.

To account for the first assumption introduced in an area weighted calculation, this
exercises uses a population weighting approach. If we look at the distribution of the population using 1970 tracts in Figure A.2, we see that the majority of the population of CG 3202 is not within the metropolitan area, but instead on the east coast of the CG. If we calculate the number of people that live within the metro area portion CG 3202, then divide by the total population of CG 3202, we will have a population weighted ratio. Figure A.2 shows that most of the area that is within the metro area of CG 3202 is sparsely inhabited or uninhabited altogether. The population of the metro portion of CG 3202 is 25,266 people. The total population of CG 3202 is 251,978 people. A population weighted ratio would then take the form of 25,266 people / 251,978 people, or 0.10.
Comparing the two ratios, 0.10 is a huge difference compared to 0.58!

We still retain the second assumption, which is that the 0.10 portion of the population that we might account for in the metro area is representative of the entire CG. Without access to the microdata or introducing new errors, this assumption remains.

To generate Figure A.1, we must project our data into WGS 84. I also chose to use a blank ggplot (\texttt{blank\_map}).

To generate FIGURE 2, we must select the necessary counties that comprise our county groups and project to WGS 84. Then we’ll calculate the density using fields that were added prior to import (\texttt{area\_sqkm}).

A.4 Analysis

A.4.1 Challenges

Two challenges were identified in the construction of the crosswalks. First, the resolution of the PUMAs and the resolution of the census tracts are significantly different. Islands in the key west, for example, have two different shapes: a rough boxy shape from census tracts, and a coastline accurate shape from the PUMAs. How then do we easily assign census tracts to their smaller geography of PUMA without accidentally assigned a census tract to its neighboring PUMA with which it might overlap in the slightest?

Second, some tracts are outside PUMAs altogether. These are tracts that no population, but are considered part of the PUMA regardless of their shapefile exclusion. Examples of this are evident in bodies of water like the Delaware Bay. PUMAs are collections of counties which extend into water and should not follow coastlines. The Delaware Bay is split between a Delaware portion and the New Jersey portion. Tracts on the east side are part of New Jersey PUMAs and tracts on the west side are part of Delaware PUMAs.
To overcome these challenges, census tracts were converted to points. Each tract was converted to its centroid point that falls inside the tract boundary. Then, cycling through each PUMA, I assign each tract a PUMA if the tract is within that PUMA. A second loop is run for each tract that was not assigned a PUMA which finds the nearest PUMA and assigns a value based on the distance to the nearest PUMA. Each of the five Census years between 1970 to 2010 were performed this way.

A.4.2 Post process MSA data

We’re only interested in MSAs with a population over 500,000 people, and we want to exclude non-MSA residuals (where msa2015 is equal to 99999). We also assigned zeros where there are NAs in the LTDB population data. Census 2010 tracts boundaries sometimes overlap with to areas that were untracted in 1970 and 1980.
Appendix B: Additional Tables and Figures

This section contains my interview protocols, figures, and tables which were not included in the text of the dissertation.

B.1 Interview Protocol

B.1.1 Informed consent letter

(Printed on Portland State Letterhead)

July 2015

Modes of Software Production and Economic Development

This research broadly investigates the software sector of Portland, Seattle and Austin. Of particular interest are the experiences of software workers in producing software, the methods you’ve used and the way your work is organized. In conducting this research, I will document the implementation and impacts of various types of software production. You have been selected for this interview because of your involvement in software production.

Your participation in this research is completely voluntary. You may suspend or otherwise discontinue your interview at any point, with no penalty. There are no anticipated benefits of a direct nature for your participation in this research.

The information you provided through this interview will be used to support research regarding software production. As such, the information you provide is confidential.
The information you provide through this interview will be referenced in a manner that would not allow a reader to identify you, directly or indirectly. In some cases, quotes or statements may be used based on non-identifying descriptors, such as “a Software Development Engineer in the Seattle region” or “a Director of User Experience in the Austin region.”

With your permission, we would like to record our interview using a digital audio recorder. The purpose of the recordings is to ensure the accuracy of my collection of the information, and to allow for a more interactive conversation. However, you may request for me to not record the interview, or certain parts of it.

If you have questions following the interview, you can reach me by email at Dillonm@pdx.edu or via phone at (503) 583-2483. If you have concerns regarding the methods used, or questions about rights as a research subject, you may contact the PSU Human Subjects Research Review Committee: by phone at (503) 725-2227 or (877) 480-4400; by email at hsrc@pdx.edu; or by postal mail at PSU Institutional Review Board, Office of Research Integrity, 1600 SW 4th Ave., Market Center Building, Ste. 620, Portland, OR 97201.

You may keep this letter for your records.

Sincerely,
Dillon Mahmoudi

**B.1.2 Recruitment message**

(Sent via email or social media)

Dear [name],

My name is Dillon Mahmoudi and I am a PhD Candidate at Portland State University, and am writing you as part of my dissertation research on the various ways software
made. I am particularly interested in how software is made in [Portland/Seattle/Austin] and how these techniques relate to regional development.

I am writing to find out whether you would be interested and available to participate in an interview to discuss your role in the software development process at [firm]. The interview would last approximately 45 to 60 minutes and would cover a variety of topics related to your current and past employment in the software sector. Your participation in this research is voluntary and confidential.

If you are willing to participate, let me know and I will work with you to schedule a mutually agreeable time to conduct the interview. If you have any questions, please do not hesitate to ask me.

Regards, Dillon Mahmoudi PhD Candidate, Portland State University

B.1.3 Semi-structured interview protocol: manager

- As I mentioned via email, I’m interested in your work in software development / software field – so I’m glad you agreed to talk today.

- How did you get into software, and then, how did you get started working at [location]?

- I’m hoping we can start with just some overview of your team/company. Can you tell me briefly what your team/company makes?

- I’m going to ask about your routine. Can you walk me through your typical day? It can be today/yesterday or any day, but what happens starting from when you wake up? Can you tell me about last week? Where you worked from? Commute times? Who you worked with and talked to?
• Who are your main competitors? Locally? What does the nature of competition look like? Do you ever work with those companies?

• What kinds of organization or associations does the firm have with other companies? What other collaborations does firm/team have with other companies? Other workers? Do you make the firm space open to the community?

• What technologies does your product build-on? Does your product use any libraries, other software products, or other code in your product?

• When hiring, what do you think makes a really good applicant for your team/company? What kinds of skills?

• What is the firm’s role in developing employee skills? Community skills?

• What’s an example of innovation in industry that sticks out to you? Can you tell me why?

• What role has the firm taken in terms of diversity?

• Thanks for your time, this has been insightful. Is there anything else you’d like to share?

B.1.4 Semi-structured interview protocol: worker

• As I mentioned via email, I’m interested in your work in software development / software field – so I’m glad you agreed to talk today.

• How did you get into software, and then, how did you get started working at [location]?

• I’ve never worked in a team like that. Can you tell me about your role in everything you’ve just described?
• I’m going to ask about your routine. Can you walk me through your typical day? It can be today/yesterday or any day, but what happens starting from when you wake up? Can you tell me about last week? Where you worked from? Commute times? Who you worked with and talked to?

• Can you tell me about your software development cycle?

• How do you track software development process, both code and tasks? What other types of tools do you use in your job to help you make software? I’m interested in everything you use – personally, as a team, and company-wide. Which of these tools do you actually use and which are you supposed to use?

• How are different features or bug fixes decided on? How is the product development decided?

• When hiring, what do you think makes a really good applicant for your team/company? What kinds of skills?

• What do you think is the hardest job in the company? Who has the easiest job here? Who could the company not do without?

• Do you have personal programming projects? When’s the last time you worked on one? What was it? Who did you work on it with? Where did you work on it? What other projects have you worked on?

• What’s an example of innovation in industry that sticks out to you? Can you tell me why?

• Can you give me an example of an innovation that you worked on within the company? How/were you recognized for that innovation?
Thanks for your time, this has been really valuable. Is there anything else you’d like to share?

B.1.5 Semi-structured interview protocol: key informant, institutional actor

As I mentioned via email, I’m interested in your work in software development / software field – so I’m glad you agreed to talk today.

How did you get into software, and then, how did you get started working at [location]?

I’m hoping we can start with just some overview of [organization]. Can you tell me briefly the role of your [organization]?

What is the competition like in software?

What kinds of organization or associations does the firm have with other companies? What other collaborations does firm/team have with other companies?

Who are the main competitors to local firms? What does competition look like locally?

What kinds of connections do firms and workers have to universities? Other associations?

What software and technologies do you hear about the most from companies? Can you give me an example? What are the most common types of software or technologies used by companies in the region? Which ones are most important?

I’m going to ask about your routine. Can you walk me through your typical day? It can be today/yesterday or any day, but what happens starting from when you
wake up? Can you tell me about last week? Where you worked from? Commute times? Who you worked with and talked to?

- What role has the [organization] taken in terms of diversity?

- In what ways do workers enhance their skills? What role has the [organization] taken in terms enhancing the skills of workers?

- What skills do you hear about the most from companies? Can you give me an example? Which ones are the most important for workers? Which ones are most important for firms?

B.2 Supplementary Figures

B.3 Supplementary Tables
**Managers. Managerial software labor market.**

**110 Computer and information systems managers.** Plan, direct, or coordinate activities in such fields as electronic data processing, information systems, systems analysis, and computer programming. Excludes occupations listed below (e.g., Chief Technology Officer, Information Technology Systems Director, Management Information Systems Director).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean $)</td>
<td>103,010</td>
<td>103,407</td>
<td>104,469</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>30%</td>
<td>29%</td>
<td>29%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.77</td>
<td>0.84</td>
<td>0.82</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>12%</td>
<td>15%</td>
<td>19%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (white, nh)</td>
<td>0.84</td>
<td>0.84</td>
<td>0.82</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Engineers/architect. High-skilled software labor market.**

**1020 Software developers, all.** Research, design, develop, and test computer applications software and operating systems-level software. Analyze user needs and develop software solutions. Design software or customize software for client use with the aim of optimizing operational efficiency (e.g., Computer Applications Engineer, Database Developer, Software Applications Engineer, Embedded Systems Software Developer, Computer Systems Software Architect).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>55,689</td>
<td>54,777</td>
<td>60,140</td>
<td>90,472</td>
<td>91,769</td>
<td>93,888</td>
</tr>
<tr>
<td>Pct. women</td>
<td>23%</td>
<td>30%</td>
<td>32%</td>
<td>23%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.72</td>
<td>0.73</td>
<td>0.83</td>
<td>0.81</td>
<td>0.84</td>
<td>0.81</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>5%</td>
<td>10%</td>
<td>11%</td>
<td>14%</td>
<td>14%</td>
<td>18%</td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.82</td>
<td>0.91</td>
<td>0.89</td>
<td>0.86</td>
<td>0.89</td>
<td>0.84</td>
</tr>
</tbody>
</table>

**1106 Network architects.** Design and implement computer and information networks, such as LAN, WAN, intranets, extranets, and other data communications networks. Perform network modeling, analysis, and planning. May also design network and computer security measures. May research and recommend network and data communications hardware and software (e.g., Network Developer, Network Designer, Computer Network Engineer).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>61,373</td>
<td>61,471</td>
<td>92,477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>26%</td>
<td>25%</td>
<td>10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.78</td>
<td>0.78</td>
<td>0.90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>19%</td>
<td>21%</td>
<td>19%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.92</td>
<td>0.96</td>
<td>0.82</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table B.1: Software occupations, historical segmentation, 1970 to 2015 (cont’d)

**Analysts/researchers. High-skilled labor market.**

<table>
<thead>
<tr>
<th>1005 Computer and information research scientists.</th>
<th>Conduct research into fundamental computer and information science as theorists, designers, or inventors. Develop solutions to problems in the field of computer hardware and software (eg Programming Methodology Researcher, Control System Computer Scientist, Computational Theory Scientist).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>71,495</td>
</tr>
<tr>
<td>Pct. women</td>
<td>14%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.67</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>5%</td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1006 Computer systems analysts.</th>
<th>Analyze science, engineering, business, and other data processing problems to implement and improve computer systems. Analyze user requirements, procedures, and problems to automate or improve existing systems and review computer system capabilities, workflow, and scheduling limitations (eg Systems Architect, Applications Analyst, Information or Data Processing Systems Analyst).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>77,585</td>
</tr>
<tr>
<td>Pct. women</td>
<td>37%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.82</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>26%</td>
</tr>
<tr>
<td>Inc. ratio (white, nh)</td>
<td>0.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1007 Information security analysts.</th>
<th>Plan, implement, upgrade, or monitor security measures for the protection of computer networks and information. May ensure appropriate security controls are in place and respond to computer security breaches and viruses (eg Computer or Network or Internet Security Specialist).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>85,700</td>
</tr>
<tr>
<td>Pct. women</td>
<td>23%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.78</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>32%</td>
</tr>
<tr>
<td>Inc. ratio (white, nh)</td>
<td>0.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1200 Actuaries.</th>
<th>Analyze statistical data, such as mortality, accident, sickness, disability, and retirement rates and construct probability tables to forecast risk and liability for payment of future benefits. May forecast future benefits (eg Actuarial Analyst, Data Scientist, Pricing Analyst, Risk Officer).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>97,080</td>
</tr>
<tr>
<td>Pct. women</td>
<td>17%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.41</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>3%</td>
</tr>
<tr>
<td>Inc. ratio (white, nh)</td>
<td>0.31</td>
</tr>
</tbody>
</table>
### Table B.1: Software occupations, historical segmentation, 1970 to 2015 (cont’d)

**1220 Operations research analysts.** Formulate and apply mathematical modeling and other optimizing methods to develop and interpret information that assists management with decision making, policy formulation, or other managerial functions. May collect and analyze data and develop decision support software, service, or products. May develop and supply optimal time, cost, or logistics networks for program evaluation, review, or implementation (eg Analytical Strategist, Business Analytics Director, Business Insight Manager, Decision Analyst, Operations Research Analyst).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>68,772</td>
<td>69,813</td>
<td>67,269</td>
<td>74,917</td>
<td>78,581</td>
<td>78,203</td>
</tr>
<tr>
<td>Pct. women</td>
<td>10%</td>
<td>27%</td>
<td>44%</td>
<td>40%</td>
<td>43%</td>
<td>44%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.67</td>
<td>0.69</td>
<td>0.75</td>
<td>0.81</td>
<td>0.79</td>
<td>0.80</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>4%</td>
<td>9%</td>
<td>15%</td>
<td>15%</td>
<td>19%</td>
<td>27%</td>
</tr>
<tr>
<td>Inc. ratio (white, nh)</td>
<td>0.93</td>
<td>0.81</td>
<td>0.89</td>
<td>0.84</td>
<td>0.89</td>
<td>0.84</td>
</tr>
</tbody>
</table>

**Administrators. Middle-tier labor market.**

**1060 Database administrators.** Formulate and apply mathematical modeling and other optimizing methods to develop and interpret information that assists management with decision making, policy formulation, or other managerial functions. May collect and analyze data and develop decision support software, service, or products. May develop and supply optimal time, cost, or logistics networks for program evaluation, review, or implementation (eg Analytical Strategist, Business Analytics Director, Business Insight Manager, Decision Analyst, Operations Research Analyst).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>75,228</td>
<td>76,473</td>
<td>74,718</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>42%</td>
<td>37%</td>
<td>38%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>18%</td>
<td>18%</td>
<td>24%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.84</td>
<td>0.87</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**1105 Network and computer systems administrators.** Install, configure, and support an organization’s LAN, WAN, and Internet systems or a segment of a network system. Monitor network to ensure network availability to all system users and may perform necessary maintenance to support network availability. May monitor and test Web site performance to ensure Web sites operate correctly and without interruption. May assist in network modeling, analysis, planning, and coordination between network and data communications hardware and software (eg Wide Area Network Administrator, Network Security Administrator, Network Coordinator).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>66,480</td>
<td>69,491</td>
<td>67,094</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>22%</td>
<td>19%</td>
<td>19%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.88</td>
<td>0.89</td>
<td>0.86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>18%</td>
<td>18%</td>
<td>24%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.84</td>
<td>0.87</td>
<td>0.91</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table B.1: Software occupations, historical segmentation, 1970 to 2015 (cont’d)

Programmers. Middle-tier labor market.

1010 Computer programmer. Create, modify, and test the code, forms, and script that allow computer applications to run. Work from specifications drawn up by software developers or other individuals. May assist software developers by analyzing user needs and designing software solutions. May develop and write computer programs to store, locate, and retrieve specific documents, data, and information (eg Systems Programmer, Computer Language Coder, Applications Programmer).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>72,905</td>
<td>74,568</td>
<td>75,301</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>27%</td>
<td>24%</td>
<td>21%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.85</td>
<td>0.90</td>
<td>0.88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>15%</td>
<td>15%</td>
<td>17%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.86</td>
<td>0.88</td>
<td>0.81</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Coders/support. Bottom-tier labor market.

1050 Computer support specialist. Provide technical assistance to computer users. Answer questions or resolve computer problems for clients in person or via telephone, or electronically. Analyze, test, troubleshoot, and evaluate existing network systems, such as local area network (LAN), wide area network(WAN), and Internet systems or a segment of a network system (eg Desktop Support Specialist, Help Desk Technician, End-User Support Specialist, Network Technician, Network DiagnosticSupport Specialist).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>52,376</td>
<td>52,847</td>
<td>52,351</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>34%</td>
<td>30%</td>
<td>26%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.89</td>
<td>0.89</td>
<td>0.88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>24%</td>
<td>27%</td>
<td>34%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.90</td>
<td>0.86</td>
<td>0.84</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1030 Web developers. Design, create, and modify Web sites. Analyze user needs to implement Web site content, graphics, performance, and capacity. May integrate Web sites with other computer applications. May use software to convert formats and use multimedia software (eg Web Designer, Internet Developer, Intranet Developer).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>47,064</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. women</td>
<td>35%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pct. minority</td>
<td>20%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.96</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table B.1: Software occupations, historical segmentation, 1970 to 2015 (cont’d)

*All other computer occupations.*

**1107 Computer occupations, all other.** All computer occupations not listed separately.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc. (mean)</td>
<td>73,630</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>66,755</td>
</tr>
<tr>
<td>Pct. women</td>
<td>11%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23%</td>
</tr>
<tr>
<td>Inc. ratio (women:men)</td>
<td>0.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.91</td>
</tr>
<tr>
<td>Pct. minority</td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32%</td>
</tr>
<tr>
<td>Inc. ratio (minority:white)</td>
<td>0.66</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.84</td>
</tr>
</tbody>
</table>