ABSTRACT

Title of Document: FLAT AND EGALITARIAN? EVALUATING WORKER HIERARCHIES IN SOFTWARE COMPANIES.

Paul Dean, Master of Arts, 2007

Directed By: Bart Landry, Professor, Sociology

The common view in the organizations literature is that, in the new economy, traditional worker hierarchies have now been replaced by flat, team-based arrangements. However, there have been few empirical studies that have tested this phenomenon. This paper seeks to fill this gap in the literature by evaluating the worker hierarchies of small and medium-sized software companies. By drawing on 61 in-depth interviews with workers and managers at 31 software companies, I assess several dimensions of organizational hierarchy. I found that worker hierarchies do not match our conceptions of traditional bureaucratic models, but formal hierarchies do remain, albeit with fewer layers. Management has relinquished decision-making on high-level decisions, while workers have gained more decision-making in production-level decisions and autonomy. I also outline the characteristics of new project-based hierarchies, which are more flexible worker hierarchies in which supervisory and managerial roles are fluid and fluctuating from one project to another.

FLAT AND EGALITARIAN?
EVALUATING WORKER HIERARCHIES IN SOFTWARE COMPANIES

By

Paul Dean

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Arts
2007

Advisory Committee:
Professor Bart Landry, Chair
Joe Lengermann
Alan Neustadtl

Acknowledgements

I especially thank my advisor, Bart Landry, for many insightful conversations and his helpful comments throughout the development of this thesis. His mentoring during this and many other projects have been invaluable throughout my years in graduate school. I also wish to thank my readers, Joe Lengermann and Alan Neustadtl, whose time and comments have greatly improved and clarified this work. Finally, much appreciation goes to Chris Andrews, Craig Lair, and Bart for use of their interview data. Works such as this one are never the product of a single individual, and it is these people who helped make this thesis possible.

Table of Contents

Acknowledgements ii
Table of Contents iii
List of Tables iv
List of Figures v
Chapter 1: Introduction 1
Chapter 2: Literature Review 5
The Social Organization of Work in the Knowledge Economy 5
Computing Culture and Hierarchy 8
The Labor Process and Hierarchical Control 12
Chapter 3: Theory and Research Questions 17
Chapter 4: Data and Methods 21
Chapter 5: Results 26
Dedicated Hierarchy 30
Project Manager 31
Team Lead/Leader 37
Developers 41
Project-based Hierarchy 45
Promoting the Ideology of Flatness 50
Summary of Findings 53
Chapter 6: Discussion 57
Chapter 7: Conclusion 57
Appendices 63
Bibliography 68

List of Tables

Comparison of Dedicated and Project-Based Hierarchies……...50

Comparing Hierarchies in Traditional Bureaucracies and Software Companies……56

List of Figures

Ideal-Typical Organization of Software Companies……28

Chapter 1: Introduction

In "The Coming of the New Organization" (1988), management guru Peter Drucker describes what new business organizations will look like "20 years hence". He compares the new organizational structure to a symphony: "A large symphony orchestra is … instructive, since for some works there may be a few hundred musicians on stage playing together … There is only the conductor-CEO and everyone of the musicians plays directly to that person without an intermediary. And each is a high grade specialist" (Drucker 1988: 48). In this metaphor, workers have specialized roles, but are not differentiated by rank and authority. They work and communicate with one another horizontally, but are not organized into a hierarchy. Vertical separation between all workers and the head of the organization is characterized by direct contact, rather than many layers of middle management.

As we converge on Drucker's prediction, it is an opportune moment to reflect on the extent of changes to organizational hierarchy and its meanings for workers. Indeed, there has been much attention on this subject in the interim. A myriad of social analysts in various disciplines have noted the flatter, more egalitarian structures of work organizations in the modern capitalist economy. Trends have been identified in the delayering of corporate hierarchies, employee empowerment, workplace democracy, and increasing worker autonomy (e.g. Piore and Sabel 1984; Jaffee 2001; Coleman 1996; Burris 1998; Rajan and Wulf 2006). These changes have unfolded in a diversity of sectors, including traditional industries such as automobile

manufacturing and the apparel industry (Applebaum et al. 2000). However, changes have been most profound in the new high-tech firms of the knowledge economy.

For obvious reasons, organizational structure is an important focus for organizational scholars and practitioners. But for sociologists, it has long been an important area of stratification research (dating back to Weber's classic study of bureaucracy). As Joan Acker (2006) argues, "work organizations are critical locations for the investigation of the continuous creation of complex inequalities because much societal inequality originates in such organizations." Greater understanding of the trends in organizational hierarchy, therefore, is particularly salient for improving our knowledge of inequality. While much evidence has established the lessening of hierarchy, it is not clear if hierarchy has been replaced by flat, egalitarian teams as many commentators suggest (Cloke and Goldsmith, 2002; Stewart, 1997; Kanter 1989), or to what degree hierarchy remains, and how it is perceived by workers.

Improving our knowledge about organizational hierarchy also has important implications for the burgeoning debates on organizational democracy and the nature of work. Stohl and Cheney argue that participation has become a "fundamental social right of people in the workplace that has value in and of itself" (Stohl and Cheney, 2001: 351). If business organizations have demonstrated an ability to maintain flat structures with high participation, we can more thoroughly assess the compatibility of democracy for profit-seeking entities (see e.g. Bernstein and Berger, 1998). Similarly, many observers conclude that the new flatter forms are more humanistic (Marchington, 1992) and have been advocated for by managers and labor alike

(Collom, 2003). Radical theorists argue that their increased autonomy, active participation, and flattened organizational structure are means for workers to recognize that management and capitalist relations are not necessary and organize themselves into a new communism (Negri, 1989/2005; Dyer-Witheford, 1999).

The pressures toward less hierarchical structures have been economic, and in some cases, cultural in nature. Flattened hierarchies are thought to increase productivity through a variety of means. In contrast to the hierarchical Fordist organizations that used Taylor's scientific management, the new organizational structures are thought to maximize worker participation and "discretionary effort" (Applebaum et al. 2000). By shifting decision-making authority to workers and enabling them to alter their routines and the production process, organizations gain by harnessing their skills and tacit knowledge while increasing motivation and flexibility.

While economic forces have placed external pressure on companies to raise productivity through organizational restructuring, cultural factors have acted internally. Workers in the knowledge economy constitute a "creative class" that is motivated by and demands greater authority in their work (Florida 2002). In software development, this "creative core" represents a distinct "computing culture" (Woodfield 2000). One of the quintessential elements of this computing culture is the raised importance of technical skills over traditional forms of rank and hierarchy. Andrew Ross, in his ethnography of a cutting-edge software company, characterized this work culture as "an anti-authority work mentality" that "over time … grew its own rituals of open communication and self-direction" (Ross 2003: 9). These

different expectations of software workers, as highly valuable employees, have further shaped the organization of work.

There is, however, reason to expect limitations to the flattening of hierarchy and the prospects for egalitarian working arrangements in profit-seeking enterprises. Writing within a Marxist theoretical framework, labor process scholars focus on the role of management in transforming labor power into labor through control of the labor process. As Bray and Littler (1988: 567) note, "this hierarchical control relation is part of the economic exchange relations of capitalism which require the constant generation of surplus value and accumulation of capital." While management is interested in empowering workers to harness their skills and knowledge, they also recognize that workers' decisions may diverge from the interests of management that seek to maximize surplus. These structural interests emerge from the relationship between labor and management and necessitates a hierarchical structure whereby management maintains control.

Given these competing pressures, it is not clear how they materialize in organizational structures and daily work experiences. Their dialectical relationship and its organizational blueprint are the focus of this study. Each of these three forces (economic, cultural, and political economic) will be explored in detail below to better determine the mechanisms at work. This will create the backdrop for the empirical study of organizational hierarchy in software firms that is to follow.

Chapter 2: Literature Review

The Social Organization of Work in the Knowledge Economy

The organization of work has changed significantly in the US since the post-World War II boom. Until the 1970s, businesses were organized in hierarchical Fordist structures that produced standardized products for mass markets. These large bureaucratic structures were characterized by inflexible career ladders and chains of command. Using Taylor's "scientific management", decisions were made higher up in the hierarchy and dictated to those blow. This signified a clear separation between conception and execution (Braverman 1974). Among workers, there was a highly specialized division of labor. Workers lost nearly all decision-making authority and jobs were highly alienating. All levels of the hierarchy were characterized by unambiguous positions of rank and authority. Workers, and their unions, acquiesced to this system in exchange for their share of the surplus in a time of rapidly rising living standards. Productivity continued to rise and business prospered.

Beginning in the early 1970s, productivity growth slowed sharply and American business faced increasing competition in an expanding world market (Harrison and Bluestone 1990). Deregulation and double-digit inflation from two oil shocks compounded pressures in the new competitive environment. Many businesses filed or were on the verge of bankruptcy, such as Chrysler, which required a taxpayer bailout to stay afloat (Applebaum et al. 2000). Firms experimented with cutting labor

costs, but failed to raise productivity. It became clear that rigid hierarchies employing scientific management were often inefficient (Jaffee 2001; Heckscher 1994). Workers disengaged from the production process were not able or motivated to share their tacit knowledge of the work. Elaborate hierarchies prevented timely flow of accurate information. Clear positions of rank and authority made the labor process inflexible and organizations were slow to adapt. Increasingly, businesses looked toward the organization of work to improve productivity, and Japan became the new model (Alcaly 2003; Jaffee 2001).

New computer technologies were also important to improving productivity. Not only did they perform labor-saving tasks, but they encouraged the delayering of hierarchies (Alcaly 2003). They facilitated horizontal communication and quicker learning across the organization and diminished the need for managerial controls. Workers regained some autonomy and were granted decision-making authority in a variety of contexts. These changes provided greater opportunities for workers to use their skills, share their tacit knowledge, and motivated them to do so. Their empowerment gave them the opportunity to alter how work was organized (Applebaum et al. 2000).

Transforming layers of hierarchy into teams opened channels of communication and facilitated further organizational changes and decision-making (Alcaly 2003). Total Quality Management (TQM) and autonomous teams became the favorite forms of restructuring (Harley 1999). These team organizations are said to be self-managing and they coordinate themselves horizontally with other teams, rather than a vertically positioned manager connecting them together. This

restructuring was often complemented with other techniques in these "high-performance work systems," such as just-in-time inventories, small batch production, incentive systems, and skill upgrading to improve flexibility and participation (Applebaum et al. 2000). This qualitative shift in the organization of production was famously labeled the "second industrial divide" and marked the transition to a Post-Fordist economy (Piore and Sabel, 1984).

While these new organizational structures and practices have been adopted in traditional manufacturing industries (Applebaum et al. 2000), they are most pronounced in high technology sectors of the knowledge economy. For example, studies have demonstrated an association between "the use of sophisticated technology and skill to the adoption of high-performance work practices" (Kalleberg et al. 2006: 279), which include flattened hierarchies and greater worker autonomy (Cappelli and Neumark, 2001; Gittleman et al., 1998; Osterman, 1994; Pil and MacDuffie, 1996; Burris 1998). Of the high tech sectors adopting the new approach, software production is particularly noted for it because of its dependence on rapid innovation and highly creative work.

These changes have led some commentators to declare that hierarchy is obsolete and that flat, egalitarian structures are the new mode of capitalist organization in the knowledge economy (Kanter 1989). Stewart (1997: 192) explains "how technology destroyed the hierarchy" and that "the flat, networked organization triumphs because the underlying economics of communication and control have changed in favor of small flexible organizations, not big ones." Cloke and Goldsmith (2002: 102) argue that "we have reached the end not only of management but of

managerial approaches to organizational change" and that we are witnessing the transformation into "organizations of, by, and for both the people who work in them and the people they serve, without distinctions based on race, gender, social class, condition of ownership, or position in organizational hierarchy." While these views are at the extreme end, they are part of a large body of literature making claim to this powerful trend toward flatness.

In this literature, these work "practices are generally assumed to have replaced hierarchical systems of control characteristic of Taylorist or Fordist production regimes by a form of work organization that empowers workers to participate in decision making, enables them to work in teams, and enhances their commitment to the organization" (Kalleberg et al., 2006: 272). However, they have largely ignored the actual existence of levels of hierarchy among high-tech workers, and relationships between these workers. While numerous scholars have noted a "lessening of hierarchy" and identified new work practices, they have missed the ways or the degree to which these organizations have flattened out and what types of decision-making has been devolved to teams of workers because of a dearth of empirical studies.

Computing Culture and Hierarchy

While economic forces have placed external pressure toward organizational restructuring, cultural factors have acted internally. Through the construction of specific norms, attitudes, values, practices, and expectations, culture can influence how organizations are shaped (Woodfield 2000). Conceptions and applications of

culture have varied widely in the social sciences and it requires clear elucidation of the cultural context relevant here (Kunda 1992). I use the concept of a computing culture here in a specific sense—one that relates to a specific of subgroup people and not in the broad context of American or western culture. In particular, I differentiate computing culture from the cyberculture studies that have gained significant attention since the late 1990s.

Cyberculture refers to "the set of technologies (material and intellectual), practices, attitudes, modes of thought, and values that developed along with the growth of cyberspace" (Lévy 2001: xvi). Empirically, cyberculture has been studied in relation to the internet (Bell 2001). Its diverse application has covered online communities (Wellman 1997; Wellman and Gulia 1999), online identity (see Turkle 1995 for the classic study), cybersubcultures (see Bell 2001, chapter 8), and other related topics. Conceptually, cyberculture is a nebulous structure. It captures many different types of people with many different types of relationships to computers and the knowledge economy. Instead, I focus on a more specific culture.

The culture relevant to this study is the social group of computing professionals who "eat, breathe, and sleep code." Like the "ethos" of Florida's "creative class", it is bounded by specific occupations (Florida 2002). These computing professionals comprise much of the workforce of software companies and computing departments in companies of the knowledge economy. They are part of the "super-creative core" of the creative class. They cohabit much of the same social space of cyberculture, but also maintain their own unique set of practices, norms, and

values, that are distinct from other cultural spaces[1]. The computing culture is much closer to the "creative ethos" of the creative class, but is even more specific.

Much of the literature and inhabitants of this cultural space define them as hackers (e.g. Himanen 2001). While mainstream media and pop culture have referred to malicious computer programmers as hackers, the traditional sense of the word is closer to those individuals inculcated in the computing culture. Even though the field has still produced a relatively modest number of empirical studies on computing culture, there are a number of consistent themes that capture its "quintessential character" (Woodfield 2000). Several authors have identified a "core ideology" of this computing culture, with common norms and values.

Perhaps the strongest element of the computing culture is their passion for their work, or their "hacker work ethic" (Himanen 2001). Hackers write and design programs because they find the work intrinsically interesting. They tend to be motivated more by this passion than the money that their employment provides (which can be significant). Challenging programs can draw the most interest and excitement, perhaps because they demand the greatest creativity. This culture encourages and rewards the most "beautiful" or "aesthetically appealing" code (Case and Piñeiro 2006).

Secondly, computing culture emphasizes knowledge sharing and open access. Computing culture has its roots in open-source software and free access. Before commercial computing became an attractive opportunity, early computer professionals survived through reciprocity and in a "culture of knowledge sharing,

[1] See Woodfield (2000, pages 9-12) for a more thorough argument of why computing may be considered a distinct culture.

continuous improving, and sharing of software" (Bergquist 2003: 225). These roots of early computing culture have been institutionally reproduced through the open-source movement, actions against censorship, and expansion of computer and internet access. Computing professionals have helped lead the broader cyberculture drive for greater access and free distribution of knowledge, through organizations such as the Free Software Foundation and the "copyleft" movement (Bergquist 2003).

The third and perhaps most pertinent cultural element to this study is attitudes against hierarchy and authority. According to Himanen, the computing culture has "always been anti-authoritarian" and "hackers oppose hierarchical operation" on both normative and pragmatic grounds (Himanen 2001: 70). Hackers are encouraged to "mistrust authority—promote decentralization" and "hackers are to be judged by their hacking, not bogus criteria, such as degrees, age, race, or position" (Levy 1984, quoted in Turner 2006: 261). Andrew Ross, in his ethnography of a cutting-edge software company, characterized this work culture as "an anti-authority work mentality" (Ross 2003: 9).

This "core ideology" of computing culture is reproduced in chat rooms, on FAQ documents, project manuals, and specialized conferences. A growing number of scholars have studied this computing culture in these various domains. For example, Case and Piñeiro (2006) studied an online programmer community that talked passionately about their coding as an art form, emphasizing coding aesthetics. Cultural reproduction in the open-source movement has gained particular attention from anthropologists (e.g. Bergquist 2003) and other scholars. It has also been

studied in more general terms at workplaces (Bloomfield 1989; Ross 2003) and schools (Sproull et al. 1984).

However, as Woodfield argues, "fewer commentators have examined the way in which elements of the wider cultural framework itself can influence and determine the nature of the immediate computing environment, or have reflected in detail upon the way the nature of the immediate context continues to surround and shape the production of the technology" (Woodfield 2000: 9). Specifically, it is not clear how this computing culture affects organizational structure or if organizational objectives are congruent with computing's "core ideology." One might reasonably expect that in conjunction with management paradigms that emphasize horizontal, team-based production, a culture that disdains authority, centralized power, and position based on rank, that software firms and workers in particular would be organized into flat, egalitarian structures.

The Labor Process and Hierarchical Control

In contrast to the flat organizational models and an anti-hierarchical computing culture, labor process theory disputes claims that hierarchies are a thing of the past, or that capitalist relations can achieve an egalitarian organization. Drawing on Marxist principles of political economy, this literature focuses on the hierarchical relationship between managers and workers in the labor process. In this relationship, the function of management is to convert workers' labor power (the potential to work) into labor (actual work effort) to further capital accumulation (Braverman 1974). Workers may resist by withholding their labor or using their labor to satisfy

their own interests, at the expense of management's (or the capitalist's) interests. "To the extent that individual or collective worker resistance interferes with it, management will be concerned to 'control' labor" (Bray and Littler 1988).

Like mainstream approaches, labor process scholars have adapted their theories to the broad organizational changes in capitalist firms. They recognize, for instance, the adoption of TQM, work teams, and increased autonomy. Of particular importance is the role of information and creativity in the knowledge economy. When managers have extended Taylorist control mechanisms to knowledge workers, such as computer programmers and software engineers, they have had poor results (Kraft 1999). These techniques stifle the flow of information and are not conducive to creative work (Andrews et al. 2005; Florida 2002). Within capitalistic enterprises "managers seeking to extract the greatest value from 'creative' workers need to manipulate not only behavior but imagination" (Kraft 1999: 21). These organizational pressures have shifted the attention away from scientific management toward construction of a workplace culture that attempts "to elicit and direct the required efforts of members by controlling the underlying experiences, thoughts, and feelings that guide their actions" (Kunda 1992: 11).

Rather than accepting these schemes of worker empowerment as new-found egalitarianism, labor process theorists point to the reconfiguration of control hierarchies. As Ezzamel and Willmott (1998: 359) note, "the intent to abolish some features of [the traditional management] system, such as close supervision, should not be uncritically conflated with the dilution or demise of an established top-down structure of control." Instead, management has reconfigured control structures to

both “control and inspire” knowledge work in these organizations. They construct a cultural environment that facilitates or “inspires” creative work, while simultaneously guiding, redirecting, and placing limits upon the work. These management interests often conflict with the workers’ interests (Applebaum et al. 2000) and their computing culture (Himanen 2001). Subsequently, “a firm’s culture may be understood as resulting from political processes where many different cultures compete for hegemony, rather than a monolithic, unified system of shared values and norms developed and promoted by management as described in the corporate culture tradition” (Rasmussen and Johansen 2005: 102), or a similarly monolithic culture of the hacker ethic. An important point is that these political processes are indeed mediated by positions of authority and ownership within the organization. The resulting control structures include specific workplace practices, organizational cultures and normative controls, and finally, (sometimes hidden) organizational hierarchies.

These competing cultures, and the organizational hierarchies they engender, can be sources of conflict within the organization. Programmers who are motivated by passion and the intrinsic interest of programming, seek interesting and challenging projects, but capitalist organizations choose projects based on greater profitability (Himanen 2001). While spaces of creativity are necessary in the labor process, the maximization and exhibition of this creativity and mastery in “aesthetically appealing” or “beautiful” code can hinder capitalistic objectives for quicker, functional product delivery. Case and Piñeiro (2006), in their study of an online computer programmer community, documented this conflict and the subversive

rhetoric and activity of participants. Participants directed their incendiary rhetoric at those with authority over them, including project managers and others having hierarchical relationships to them. Non-technical superiors are particularly common sources of conflict (e.g. see Lewis 1999; Zmud 1982; Burris 1998). These conflicts may be further compounded by the tension between creating software with free access versus proprietary software for maximum profit.

Recent qualitative studies have identified new strategies of control in managing the labor process of knowledge work. For example, several studies have shown that by using "soft control" (Florida 2002), workers "who are offered autonomy over their work, including autonomy over their working time, are motivated by this (Bailyn, 1988) and are willing to work long hours (Barrett, 2005; Voss-Dahm, 2005)" (Rasmussen and Johansen, 2005: 102). Others have explored how identity (Marks and Lockyer, 2005) and organizational policies (Baldry et al., 2005) facilitate control and the extraction of greater value. However, these practices exist within organizational structures that have not been sufficiently explored. In much of the literature, organizational structure and the labor process are typically conflated (Ezzamel and Willmott, 1998), but demand a greater focus in light of current debates. Almost no empirical studies have assessed the flatness of organizational hierarchy and critical scholarship has focused on the newer modes of cultural control (Kunda 1992), while ignoring modes of control exercised through organizational structures. When researchers have assessed the "flat thesis", they have focused exclusively on managerial hierarchies (Rajan and Wulf 2006), thereby

ignoring the organization of worker hierarchies. A more nuanced analysis of organizational hierarchies may reveal increasingly complex stratification structures.

Chapter 3: Theory and Research Questions

This study seeks a grounded approach toward new forms of organizing work in the post-industrial economy, focusing on software firms as exemplary of the knowledge economy. While much of the literature has described it as a vague "lessening of hierarchy", many others have made more bold (but often less empirically rigorous) arguments. They argue that new horizontal coordinating mechanisms such as teams (Marchington, 1992) utilizing TQM and other HPWP (Applebaum et al., 2000) have "destroyed the hierarchy" (Stewart, 1997) and ushered in new workplace democracies (Cloke and Goldsmith, 2002). Managers support these organizations because of greater efficiency, and workers inculcated in the computing culture identify with the decentralized "hacker ethic" (Himanen, 2001) and its "anti-authority work mentality" (Ross, 2001: 9). This study asks are these organizational structures as flat and egalitarian as they are thought to be?

Much of this literature implicitly and explicitly conceives of this transition as an act of replacement (e.g. Kalleberg et al., 2006). As hierarchical structures diminish (approaching absolute dissolution), horizontal coordinating mechanisms are developed proportionally. This transformation is subsequently interpreted by scholars and practitioners as democratic and egalitarian. However, the literature errs by conflating "processes of coordination … with structures of control" (Ezzamel and Willmott, 1998: 362). This conceptual distinction is of utmost importance to this study. The relationship must be seen as processes of coordination *embedded* within

hierarchical structures of control. For example, teams of empowered workers may be organized horizontally, but be assigned to specific projects conceptualized by those above them. Workers may be free to determine how tasks are carried out, but work within prescribed parameters. Furthermore, these hierarchical structures may be characterized by varying degrees of formality and informality.

Critical scholarship from within Marxian labor process theory has illustrated this distinction. Rather than emerging from a natural and functional necessity, these hierarchical structures remain because of the structural imperative of capitalist organizations. Workers' interests may be structurally different (Ezzamel and Willmot 1998; Applebaum et al. 2000: 8) and often overtly opposed to the interests of management. To the extent that this is the case, managers construct both normative and organizational controls to ensure that their interests are met. These controls may result in both formal and informal organizational hierarchies. To evaluate these hierarchies (or lack thereof) in software firms, I am concerned with three separate but highly related questions.

First, *what formal positions do workers occupy in the organizational structure?* Titles differentiate workers from one another, but organizational contexts attribute ranking and ordering of these positions. The focus here is on how workers are arranged, relative to one another (i.e. who reports and is responsible to who). Are workers arranged horizontally in teams with direct reporting to a CEO (similar to Drucker's symphony metaphor)? Or are they arranged in vertical hierarchies and interface with middle management (bearing resemblance to traditional bureaucracies)? By workers, I mean those individuals that directly execute the design

and development of software products. I do not evaluate managers who focus exclusively on managing developers or the organization. One could similarly look at the management structure, but this is beyond the scope of this study.

Second, *how is decision-making authority distributed among workers?* Since Weber, authority over decision-making has always been a key attribute in the study of hierarchy and bureaucracy. In a hierarchical organization, decisions are made from above and dictated to those below. In this way, workers receive orders rather than make decisions. In flat structures, all members participate in decision-making, perhaps as individuals, or in groups through voting and committees. These may range from high-level decisions such as budgeting, hirings and promotions, and setting policies and procedures, to lower level decisions including task assignments, setting deadlines, and task execution. It is often taken for granted that workers do not participate in strategic company decisions, but this remains an important component of hierarchical control and inequality (Acker 2006). Higher-level decision-making is also still very important for workers because such decisions have daily impact on workers and the organization of work. It is logical to assume that the degree of flatness will coincide with the devolution and sharing of such decision-making. I follow Wright et al. (1995) in viewing decision-making authority as separate and unique from offering input and advice.

Third, *do workers experience upward mobility in their organizations*? As Barley and Kunda (2001: 87) state, "hierarchy not only defines the distribution of authority in a bureaucracy, it provides a blueprint for constructing organizational careers." Vertical mobility in an organization requires multiple levels in a hierarchy.

Generally, but not always, such a move will result in greater decision-making authority and supervisory/managerial responsibilities over subordinates. This is contrasted with horizontal mobility, where workers may attain new formal positions with new specialization, but without new authority and power. This latter movement could occur in either flat or hierarchical structures, and is therefore not indicative of either. It is not a focus here.

These three questions seek to paint a more nuanced picture of hierarchy among workers in knowledge economy firms. They strive to capture the objective existence of organizational structure in software companies. This will subsequently improve our understanding of stratification within organizations.

The nature of this study lends itself to an exploratory analysis. While literatures on the social organization of work, workplace democracies, and computing culture suggest pressures toward flatness, labor process theory that hierarchy and complex inequalities remain despite these pressures. It is unclear how these forces will bear themselves out in the software companies that are the focus here. Answers to my three research questions can only be found in rigorous empirical analysis in hopes of contributing to these lively debates.

Chapter 4: Data and Methods

This analysis draws on in-depth interviews with 61 respondents at 31 different software companies. All companies represented in the interviews are considered small to medium sized businesses (based on US standards). At most companies, two or three interviews were conducted, including the founder and/or CEO and one or two programmers.

All interviews were conducted in-person in the summer and fall of 2001 using semi-structured questionnaires. Interviews took place in the Washington, DC/Baltimore metropolitan area, home to the third largest concentration of software firms in the US. Initial contacts were identified through personal acquaintances and friends. Subsequent respondents were found through referrals to co-workers, friends, and acquaintances using snowball sampling (Weiss, 1994). Additional respondents were recruited by identifying potential subjects in newspaper articles, news releases, and web-based directories. These people were then contacted via phone or email requesting interviews with the CEO and a software developer or similar position. Out of a total of about 100 companies that were contacted, interviews were conducted with approximately 31. The remaining companies either declined to participate, failed to respond, or postponed participation until a later date.

While this data is not representative of all software companies, efforts were made to construct an appropriate sampling frame. Companies ranged in size from a startup with a few individuals to established companies with several hundred

employees at offices spread around the US. The smallest companies had only two to four full-time employees, but many hired part-time freelance programmers. The largest companies occupied offices on multiple floors in high-profile office buildings. Some of these companies also had satellite offices in Europe or had recently closed them. However, given the timing of this data collection (the dot-com bubble burst in March of 2000 and effects were continuing to be felt through 2001 and beyond), personnel numbers were not entirely stable. In particular, many companies were downsizing and some smaller companies were struggling to survive. Companies also varied in their location within the Washington, DC/Baltimore metropolitan area. They were spread around downtown Washington, Baltimore City, suburban Maryland, and the Virginia high-tech corridor. Finally, individual respondents varied in terms of their position within the organization, title, and level of responsibility. These included senior and non-senior management, workers of various titles, and both technical and administrative positions.

Interview questions covered titles, mobility, company structure, occupational roles, the labor process, and company culture. The Appendix provides sample questions on the questionnaires administered to programmers and technical leads/project managers respectively. Sometimes it was only possible to ask specific questions of a single person in a company. Other times, the same questions were asked of multiple people, allowing checks for consistency. The majority of the interviews took place at respondents' offices. They averaged one to two hours in length, and all interviews were transcribed for data analysis. The interviews were conducted by a member of the university faculty and four graduate students. The

author did not conduct the interviews and therefore transcripts are treated as secondary data.

The exploratory nature of this research lends itself to an inductive approach. During the course of data collection, the use of a semi-structured questionnaire permitted modifications as new themes arose or specific topics required further probing. This research design incorporates a grounded theory methodology, which emphasizes theory construction from data (Glaser and Strauss 1967; Charmaz 1983). This is further employed throughout the data analysis.

Consistent with grounded theory principles, themes were first identified in the interview data. By first reading through interview transcripts, I constructed several themes related to hierarchy in software companies. I then developed a list of search terms that corresponded to each theme. Using the software package Atlas/ti, I systematically searched all 61 transcripts for each instance of the search terms. The search terms enabled me to identify relevant quotes/phrases and based on these quotes, I acquired new search terms that were then used to continue my search (search terms were also obtained from the literature). In grounded theory, this process of linking quotes to emergent themes is called coding.

As the coding process continued, the software package enabled me to merge or delete redundant codes, while new codes could be further assigned to emergent themes. This coding occurred concurrently with readings from the literature in an on-going process that integrates the data with existing literature. During this process of triangulation between data, codes, and the literature (Charmaz 1983), I constructed

several memos that facilitated the inductive development of new concepts. The software's functionality permits such memos to be connected to these codes.

When all coding was completed, I was left with a refined list of relevant codes with many quotes connected to each code. However, these quotes represented both supporting and disconfirming evidence for each particular theme. To evaluate the strength of each theme, I constructed a data matrix (Miles and Huberman 1984). This matrix consisted of columns of supporting and disconfirming evidence for each theme and each respondent. The matrix served as a visual aid to identify the relative support for each theme and facilitated the identification of clusters of relevant data. This technique prevents making claims on scant or ambiguous evidence.

The use of interviews with both workers and management is a unique facet of this study. Not only did this allow me to generate better images of work organizations through multiple perspectives, it provided valuable insights from individuals performing the actual development work. While most studies exploring organizational structure, coordination mechanisms, and HPWP have used surveys answered by human resource administrators and managers (e.g. Osterman 2000; Eriksson 2001; Caroli and Van Reenen 2001), these miss the perspective of the workers themselves (Applebaum et al. 2000). Such one-sided views may not necessarily reflect worker relationships, decision-making, and mobility opportunities, and are greatly enriched when complemented by interviews with workers. While a handful of ethnographic studies (Ross 2003; Woodfield 2000; Barker 1993) have captured workers' perspectives in teams through in-depth interviewing and observation, these studies have tended to focus on a single company. Furthermore,

most have only marginally considered organizational structure and authority (Ross 2003; Woodfield 2000).

Chapter 5: Results

The results reported here test the "flat thesis" by focusing on worker hierarchies in software companies. To evaluate worker hierarchies in this study, I focused on three primary questions: 1) what formal positions do workers occupy in the organizational structure?; 2) how is decision-making authority distributed among workers?; and 3) do workers experience upward mobility in their organizations? I have constructed a typology of worker hierarchies, which weaves together evidence that addresses these questions.

In order to fully appreciate the relative positioning of worker hierarchies, it is also helpful to understand the broader organizational context within which these worker hierarchies are situated. Therefore, I begin reporting results by briefly looking at the organization as a whole. It is my intention that this holistic approach will help our broader understanding of the depth between workers and the top of the organization, and keep us conscious of the decisions affecting work which remains entirely with management. This introduction will set the stage for evidence presented on worker hierarchies, which comprise the focus of this analysis.

As stated earlier, I differentiate between workers and managers in this study. By workers, I mean those individuals who directly execute the design and development of software products. I do not evaluate managers who focus exclusively on administrative and managerial tasks. These full-time managers comprise a managerial hierarchy that exists above all workers in the organization. For example,

consider John[2], who worked as the Vice President of Product Development at a software company. In a position of middle management, he saw hierarchy above and below his position.

> When it comes down to it, there's a clear chain of command. I know exactly what I'm – [the CEO] tells me exactly what I'm responsible for. And then I delegate that and tell people what they're responsible for.
>
> (John, VP of Product Development, Company D)

John's comments reflect a common observation in software firms. Not only did a chain of command exist, but individuals understood it and their position within it, if only subconsciously. Furthermore, these hierarchical relations existed among both managers and workers themselves.

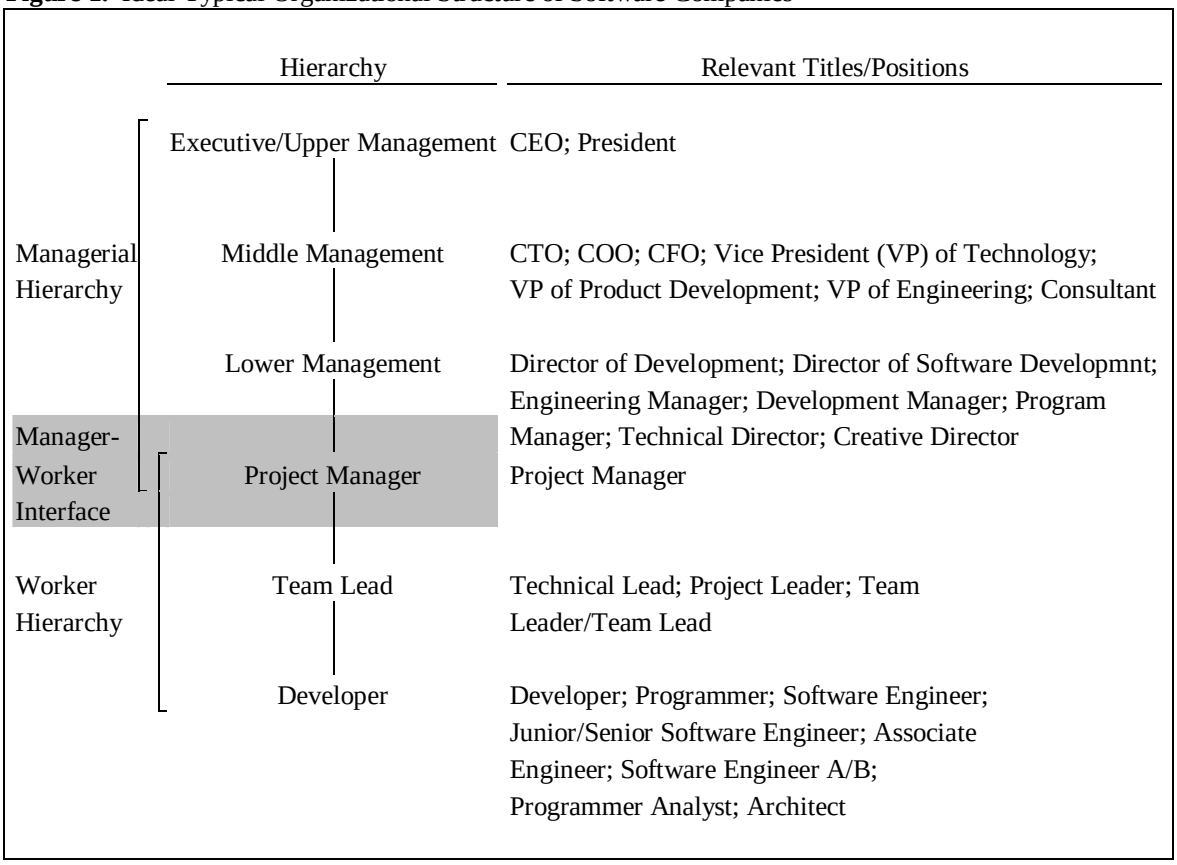
By drawing widely on interview transcripts, I was able to trace the structure and flow of these chains of command[3]. When I compared these organizational structures, I found some variation existed between organizational hierarchies, particularly in the number of layers of management, which generally varied with company size. The fact remained, however, that all companies did adopt a managerial hierarchy. Furthermore, there was considerable continuity in the positions and organization of worker hierarchies and the worker-management interface.

Based on these chains of command, I constructed an ideal-typical organizational chart of software companies (I have chosen not to look at other departments, such as Sales or Human Resources, and focused only on Software

[2] Actual names of all respondents and their companies have been removed to maintain confidentiality. Pseudonyms have been used in their place.

[3] Using available information, I was able to reconstruct organizational charts for half (16 of 31) of the companies in the sample. I lacked information to reconstruct an adequate chart for the remaining 15 companies, but I did gather detailed information on the worker hierarchies within most of these firms.

Figure 1. Ideal-Typical Organizational Structure of Software Companies



Development). This chart is represented in Figure 1. At the top of the hierarchy is the management hierarchy, which I represented simply as Executive/Upper Management, Middle Management, and Lower Management. I have also identified the titles that correspond to each layer of management. No organizations had all titles that are listed and managerial hierarchies normally had two or three, and sometimes four layers. For example, at Company K, the CEO sat atop the managerial hierarchy, with the Vice President of Engineering reporting to him, and the Director of Development, and Project Manager, at respectively lower levels below. At Company N, the Project Manager reported to the Technical Director, who reported to the President.

In most organizations, the Project Manager was the interface between workers and managers. I have included it in both the worker and management hierarchies because the Project Manager often performed development duties, in addition to his/her managerial responsibilities. This dynamic is explored in greater detail below. Under the Project Manager is the Team Lead, and finally developers are at the bottom of the hierarchy. Most organizations had both a Project Manager and Team Lead, but this was not true in all cases. However, even in their absence, respondents often referred to them as typical positions in the industry. Developers had the most variety in titles within their position (e.g. Associate Engineer, Programmer, Junior/Senior Software Engineer, etc.). While some of these positions implied hierarchy (i.e. Junior/Senior Software Engineer), there was no differences in formal decision-making authority or supervisory relationships within these positions and they have been subsequently grouped together. Many developers reported that such distinctions had no meaning ("titles don't really count too much" (Anad, Developer, Company I) or "titles in the company mean nothing" (John, VP of Product Development, Company D), but they did tend to reflect differences in skill levels, experience, and often status.

The existence of hierarchy was nearly universal[4] in the software companies studied. Even small companies adopted minimal hierarchical forms that had permanent managerial and administrative positions with superordinate positions over workers designing and developing software products. As stated previously, my focus

[4] One company, Company G, did appear to be truly flat. However, this company had only three individuals who created the organization as a limited liability partnership. Karl, a Founder and "Principle" of Company G, reported, "in order to be three equal partners, we're sharing the decisions, and sharing the revenue." He and his partners avoided arranging themselves hierarchically because "a hierarchy, by definition, is not that" (that being the equal sharing of decisions and rewards).

is specifically on the worker hierarchy within these organizations. I found that the organizational structure of these *worker hierarchies* can be classified into two general and distinct hierarchical forms. First, there are dedicated hierarchies with relatively static vertical relationships that reflect many of our traditional notions of organizational hierarchy. Second, there are more dynamic project-based hierarchies, where selected workers temporarily assume superordinate positions, and are subsequently de-activated from their position and replaced by another worker as projects reach completion. Each of these forms are explored below.

The images of organizational work charts given above accurately portray formal positions and implicitly assume delegation of authority and power relationships. However, such assumptions do not necessarily reflect daily work practices. To understand the nature of the relationships, we must "bring work back in" (Barley and Kunda, 2001; Barley, 1996) by evaluating worker experiences and meanings. Within these positions, how is decision-making authority distributed? Do workers experience vertical mobility between positions? Answers to these questions are woven together to construct an understanding of the two hierarchical forms (dedicated hierarchies and project-based hierarchies). Accordingly, the focus shifts from the abstracted role of various positions to the concrete experiences of workers and managers occupying those roles.

Dedicated Hierarchy

In a dedicated hierarchy, superordinate and subordinate workers maintain relatively static relationships, relative to one another. The positions have clearly

understood boundaries; titles tend to reflect and communicate such boundaries. These boundaries and vertical relationships are understood within the three positions of the worker hierarchy and worker-management interface: Project Manager, Team Lead, and Developer.

Project Manager

The Project Manager's duties are diverse. Just some of the Project Manager's responsibilities include communicating between clients and the development group, interfacing with management, sales, and development, drafting requirements documents for new projects, and coordinating the design of new projects. However, the position of Project Manager cannot be viewed as simply a managerial position. Rather, in many companies, large and small, the Project Manager is directly involved in designing and coding of software products. In other words, they both managed the software development process and developed the software itself, but they did tend to dedicate more of their time to managing. The extent to which their tasks covered either managerial or development duties varied both across and within organizations. Accordingly, the Project Manager position ambiguously represents both the worker-management interface and the pinnacle of the worker hierarchy. For instance, Syd is a Project Manager that does coding occasionally, but focuses mostly on managerial responsibilities.

Question: Did you do any developing and actual coding as the Project Manager?

A bit, on some of the smaller ones I had to lend some coding help. That was mainly for projects that were less than a month in scope, two to three resources. The big ones I haven't.

(Syd, Project Manager, Company C)

In other (often larger) companies, Project Managers may "just mange perk charts and everything, and they don't even know anything about programming" (Kevin, Team Lead, Company M). In addition to variation across companies, there was also differences among Project Managers within companies. Trey is a Developer who has worked for both types of Project Managers.

There are project managers who … are close to senior software engineers, and the project managers that are closer to being just plain managers.

(Trey, Developer, Company K)

Regardless of the distribution of coding and managerial duties, Project Managers required certain levels of technical skill. This is evidenced in the fact that many Project Managers interviewed were actually upwardly mobile developers and architects. In some cases, such as John, Developers were promoted to Project Manager, and continued their climb through the managerial hierarchy.

Question: What position did you come on as?

A developer.

Question: So you went from developer to—

Project Manager, to Director of Operations for the East. I used to run Operations for the East. From there, I went to—I had a lot of success at sales and starting new offices … so they had me switch over to new office

> development, and I managed all the new offices … and then, from there, VP of Product Development.
>
> (John, VP of Product Development, Company D)

Nonetheless, a few companies had professional managers with only supplemental technical training. However, many developers spoke negatively of non-technical project managers, which confirms previous findings (Zmud 1982).

Despite responsibilities in coding and software development, Project Managers maintained superordinate positions, over other workers (i.e. Team Leads and Developers). Project Managers reported directly to managers higher up in the hierarchy, such as a Director of Engineering or Vice President of Software Development. They were responsible for individual projects and the individual workers assigned to those projects. Syd, the Project Manager who stated earlier that he does occasional coding, made this relationship explicit.

> Basically you have a fairly hierarchical vertical structure. You're supposed to have every project manager managing a project, every project manager reporting to a director.
>
> (Syd, Project Manager, Company C)

As part of the management team, Project Managers reported participating in higher-level decision-making functions. For example, they determined the number of "resources" (Developers) necessary for a project and in some companies, they assisted in developing the budget. Developers and Team Leads rarely participated in these decisions. Project Managers also usually set deadlines, and while the scope of

individual pieces were negotiated between Developers and managers, Project Managers made the final decisions.

While the team lead was most often the position directly below the Project Manager, the Project Manager also assisted in the management and supervision of the Developers. This management relationship included a number of directives and means of control over the work of Developers. Ahmet, the Co-founder and CTO of one company, identified their Project Managers' role in task assignment.

> The project manager really has the authority and the judgment to say "Hmm, I've done a skills assessment and I think this person is better capable of doing this than this person."
>
> *Question*: Do the members of the team have some say in what they're assigned to?
>
> They do. Sometimes you get volunteers, saying "hey, I really want to work on this." Sometimes you have to play Spock, and you're to say "Sure, I know you want to work on this, but in the interests of time and the fact that we don't want a lot of R&D, I'd like X, Y, Z to work on it because, you know, they've really done it before."
>
> (Ahmet, Co-founder/CTO, Company L)

At Company L, Project Managers did not exhibit direct control of assignment of tasks. Instead, Developers were given opportunities to volunteer for and give input on tasks they wanted. However, it was the Project Manager who got to make the final decision. The assignment process was done informally, with the goal of seeking consent, but was an unobtrusive means of control mediated by this higher position. In

fact, this authority was common across many organizations and decisions. A Project Manager at Company J, Jack, reiterated this authority.

> *Question*: Is there one person who's kind of in charge of [task assignment], or is it more of a communal offering?
>
> I have the final say on kind of everything that happens, so I put myself at the top.
>
> (Jack, Project Manager, Company J)

According to Jack, he had the final say on "everything" with Developers. Other areas of decision-making attributed to Project Managers included setting deadlines, the level of documentation necessary in coding, hiring, firing, and promotions. At Company D, Project Managers were also responsible for monitoring the actual coding and selecting relevant features in a software product.

> One of the biggest challenges for a Project Manager, and something that I'd like to think I was always good at—developers have this inherent quality, if you will, where they want to build a monument to themselves. And they introduce complexity, because, "It would be really cool if I could do this thing. And it provides me with all of this future ability to do something the customer's not asking for." So as a Project Manager, as a manager in any respects, you look for those people who like to build a monument to themselves. You keep an eye on those people. And I used to come back to them, say, "It's got to be simpler than that."
>
> (John, VP of Product Development, Company D)

This struggle between Developers and Project Managers (and to a lesser extent, Team Leads) was a common occurrence in these software companies. Developers often favored writing "elegant code" that was "cool." We know from recent studies on computing culture (Himanen 2001; Case and Pineiro 2006) that the writing of such "beautiful" and "aesthetically appealing" is encouraged within the profession. In fact, status rewards can accrue to those who have demonstrated such coding skills. However, managers, including Project Managers, have different priorities. They are under constant market pressure to make productive use of their resources (including Developers) and the time necessary to write such code (or "monuments") comes at the expense of other projects and tasks. Accordingly, this hierarchical authority over workers serves the interests of the founders and owners to improve the surplus of software development.

These comments have demonstrated that Project Managers occupy a formal position between the worker and managerial hierarchies, while existing in both simultaneously. They have some authority and input in high-level decisions, and significant decision-making authority in production-level decisions. In most organizations, the Project Manager yields significant decision-making authority over Team Leads and Developers, but this is not always the case. Some organizations, such as Company C, view this position as more of a coordinating position.

> The Project Manager owns this whole thing and brings in other people, if necessary. [But] the team model is based on a team of peers. And so that's where you have, even though the project manager is responsible for this, the project manager isn't the boss guy.

(Roger, Founder/CEO, Company C)

Accordingly, it is important to keep in mind the variation among organizations in distributing power and authority throughout the hierarchy. Indeed, a few organizations did not have anyone in this position. Nonetheless, Project Managers in dedicated hierarchies tended to be technically-trained individuals in formal hierarchical positions and given decision-making authority over workers and participation in broader organizational processes.

Team Lead/Leader

Team leads generally report to the Project Manager, or Director of Development/Engineering, if a Project Manager is not present or nonexistent. In dedicated hierarchies, they also blended managerial and development responsibilities. But in comparison to Project Managers, they focused more on coding and development and less on general and higher-level managerial tasks. In dedicated hierarchies, Team leads were all Developers and designers at some point in the past (and in certain ways, they still are) and usually had achieved their position through formalized promotion processes. This was the case for Thom, who started as a senior software engineer and was recently promoted.

> This year, um… I get promoted to be Technical Lead.
>
> *Question*: What will be the change in responsibilities that will go with that?
>
> When you work as a software engineer, pretty much you're responsible for your own developments … But if, um, you become Technical Lead, then you have much broader responsibilities that you have to work with, a lot more people … it's more like, uh, a lot more to do with management and control.

(Thom, Team Lead, Company B)

As Team Lead, Thom gained a managerial, or supervisory position, among the developers which gave him more control. He became responsible for both his own work and the work of his team. When asked if there was a lot of upward mobility at his company, Thom indicated that there was.

This additional responsibility was normal for Team Leads in this type of hierarchy. This added responsibility and control made Team Leads feel and appear as quasi-managers. This point was made by Kevin, a Team Lead, and confirmed by the CEO of his company, Company M.

> A lot of companies, larger companies, what they do is they divide the work between the manager, who just mange perk charts and everything, and they don't even know anything about programming. Sometimes it's difficult to work in that environment, but you know, to each their own. And then other times, you know, they have a senior engineer, and they're the ones that are actually making all the decisions and doing what I would say is the real work inside there, and here at Company M, we have a team leader position, which is a combination of the both, which...
>
> *Question*: Your team leader is also a part manager?
>
> Well, that's what I'm saying. It's the best of both worlds, in my opinion.
>
> (Kevin, Team Lead, Company M)

> Now a project leader is, I mean, it’s kind of a management position, but, I mean, the project lead is also writing code as well.

(Michael, CEO, Company M)

With new responsibilities, Team Leads also attained greater authority over the Developers in their team. They were assigned more decision-making authority in a variety of situations. In traditional hierarchical organizations, this authority is normally granted only to managers. In fact, many of these areas overlapped with the authority of Project Managers. For example, Team Leads, such as Kevin, were often responsible for decisions on task assignments.

> I'm the team leader. I'm responsible for all of my guys, and if they're not, if they're not doing that, it's my problem, not theirs … As a team leader you have to be able to understand the strengths and weaknesses of everybody on your team so you can decide who's gonna be working what task and where they're going to be placed to best … accomplish the goal, if you will. And that's pretty much how it works out.
>
> (Kevin, Team Lead, Company M)

In addition to task assignment, Team Leads often set deadlines and maintained project schedules.

> *Question*: How did you all decide on what portions you all would get to work?
>
> Let the team lead take care of that. She asked us, "Do you have anything in particular that you want to work on?" But the project schedule and who was working on what was all decided by the team lead.
>
> (Salih, Developer, Company E)

Because they were responsible for their Developers, Team Leads were concerned about their the Developer's productivity. They monitored individual member's progress and could sometimes sanction team members for not meeting standards or objectives.

However, it was rare for Team Leads to be involved in high-level decision making such as company strategy, creating budgets, resource allocation, drafting policies and procedures, etc. These were normally reserved for full-time managers and sometimes, Project Managers. Occasionally, however, Team Leads were given higher level authority. For instance, Team Leads did have some decision-making over hiring.

> The project leaders look at the resume, if the resume really looks interesting we'll bring the person in and as part of bringing the person in, they sit down with them, grill 'em about what they've done … then we usually take the person out to lunch and whatever the team they'd be working with and then a few other people from some of the other disciplines. You know, we all throw rocks at each other and have a good time and then uh, everybody writes up their impressions and staples them together and gives those to the project leader, project leader makes an assessment from that, and I either give the thumbs up or the thumbs down and I never question those recommendations because at that point, you have to trust the process.
>
> (Michael, CEO, Company M)

In this case, Michael, the CEO, had formal decision-making authority over hiring new Developers. On the one hand, his Team Leads only made a "recommendation" on

new hires, but on the other hand, the CEO always trusted and reiterated their recommendation. His rationale was that "because even if you bring somebody in over that kind of recommendation, then what are you saying to your existing employees?" Accordingly, recommendations took more the form of decisions than advice.

It became clear that many Team Leads understood their position as partially managerial, in a vertical relationship to Developers. Successful developers, like Thom, were promoted into the position, thereby receiving additional responsibility and decision-making authority. In many ways, however, the position was spoken in terms of coordinating among teams or being a contact person during projects.

> The majority of the responsibility that qualifies them as a lead, they're going to be the ones that coordinate between other groups and, you know, gather requirements or discuss what will be necessary from Group A to work with us and talk to Group C when they have a request for an enhancement or something like that.
>
> (Salih, Developer, Company E)

Therefore, the Team Lead should not be thought of entirely as a vertically hierarchical position, but one that also coordinates horizontally. Because the Team Leads also do coding and executing the technical design, they do work alongside Developers in a way that managers would not.

Developers

Developers work at the bottom of the hierarchy in software companies. Unlike full-time managers, Project Managers, and sometimes Team Leads, they

almost never have decision-making authority concerning high-level affairs or authority over the concerns of other workers. While some are able to work their way up into the managerial hierarchy, many workers lack the requisite managerial training, and are therefore often excluded from positions of power and authority within the organization.

However, developers do not fit our traditional notions of powerless workers in pyramidal bureaucracies. Despite being in dedicated hierarchies, they do have a level of control and authority over production-level decisions and aspects of their own position that other types of workers lack. As professionals, these advantages cover a variety of aspects of their jobs. For example, Developers have task autonomy, where they can exercise discretion in determining how best to complete a task, or set of tasks. Eddie, the Director of Development at Company A, makes such a point.

> You have a certain amount of freedom in the things that you can develop here. You know, you don't have… big brother, for example, telling you what everything is gonna have to look like and act like. You have a certain amount of freedom to do what you like. As long as it meets the core requirement of doing whatever it's supposed to do right.
>
> (Eddie, Director of Development, Company A)

Regardless of one's position in the hierarchy, they almost always have time autonomy at their jobs.

> The hours are what we make them. It depends. Nobody's punching a clock here. Generally we come in sometime between 9:00 and 10:00. And we're

here until some obligation has pulled us away, or we decide that we've had enough of the office and need to go home and eat, or whatever.

(Clayt, VP Engineering, Company F)

These findings are congruent with other research on software workers and professionals in general.

For some Developers, however, this is not enough for them. There were times when developers felt that information about the company's status or direction were being held from them. At times, there were decisions made within the organization that effected them, for which they felt they had no control over. For example, Jessica enjoyed control over her own position, but was upset with not having decision-making authority on hiring.

Question: How much control would you say you have over the decisions that affect your position?

… I think I have good control over that. What I feel I don't have control over is like we're hiring all these … visas, and I'm like, "Hello, when are we going to stop doing this?" And I keep saying to [the co-founder], "We're not getting quality people this way" And [the co-founder] goes, "Well, we're getting them at a pretty good rate." I'm like, "Well you get what you pay for." You know, like, "Look at the situation here. When are we going to stop this?"

(Jessica, Developer, Company L)

Despite this power differential, there appeared to be little conflict or resistance. In actuality, some developers did not indicate dismay at their position's lack of authority in the organization. Many developers were interested only in the development

aspects of the business, and they indicated they would not want a managerial position because it would give them a different focus than developing. In doing so, they normally accepted the unobtrusive commands of management.

On the other hand, many Developers were interested in attaining greater authority and managerial responsibilities in their companies. Rather than hoping management would devolve these responsibilities downward through delayering, they sought to climb the organizational hierarchy individually. Because of the number of layers in the organization, they perceived vertical mobility opportunities through the hierarchy. While he was still a Developer, Eric envisioned such a path for himself.

> There's a lot of different positions and different levels within the company, lots of opportunities, things going on. So I would like a career with a company, rather than just to move around every couple of years.
>
> (Eric, Developer, Company D)

Smaller companies often had less opportunities for vertical mobility, but Developers did not always see this as a permanent limitation for their upward mobility. Instead, they envisioned organizational growth that would allow them such access to a blend of development and managerial tasks.

> My titles probably changed because we are supposedly growing, you know, we are growing, we're hiring more people, and, um, eventually I will be doing more management, versus just execution of the tasks.
>
> (Nam, Developer, Company P)

In sum, Developers did not entirely resemble workers in traditional bureaucracies. They possessed greater (but still limited) production-level decision-making and

individual autonomy, but like Team Leads and Project Managers, they still lacked decision-making and even input on higher-level decisions. They did perceive vertical mobility opportunities through these respective positions, commensurate with levels in the worker hierarchy, and even possibilities of mobility within managerial hierarchies.

Project-Based Hierarchy

In each of the positions of a dedicated worker hierarchy, the individuals occupying those positions were stable over time. That same individual occupied that position from one project to the next, until he or she would leave the company or get promoted into another "permanent" position. When Thom got promoted from being a "Senior Software Engineer," his title and responsibilities changed, then becoming "Technical Lead." In other words, he became *dedicated* to that position, and once he attained that position, his role remained stable and consistent over time. After time there, he may have been promoted to Project Manager, or another higher position, or eventually left the company, but he probably would have never returned to his position as Developer. This is certainly no profound observation, as it matches our traditional notions of organizational hierarchy. The interesting part, however, is that not all of the organizations sampled fit this profile.

Instead, some organizations adopted what I have called a project-based hierarchy. Workers could be assigned, removed from, and re-assigned into lead positions from project to project. Unlike in dedicated hierarchies, when developers were assigned as Team Leads, their title did not change and this was not viewed as a promotion. One developer, Salih, described this fluidity.

> I'm not sure what [title]'s written on [the lead's] business card right now. I know people whose title have changed six times, and it's not because their responsibilities have necessarily changed that much. … There's someone who's acting as a lead right now because of a project that we're working on, but not in an everyday sort of sense.
>
> (Salih, Developer, Company E)

In project-based hierarchies, the Team Lead was not a permanent position and different Developers served as Team Leads on different projects. During the project, they particularly acquired greater authority in production-level decisions. Anad made this similar point.

> To be frank, titles don't really count too much [laughter]. I've pretty much had, uh, the same set of responsibilities of maybe about, less than a month…in a month's time, I had the same set of responsibilities that I had then till now. Its just been growing, as things change and things improve and uh, I pretty much take up the lead in designing and uh, conceptualizing and strategizing products. Uh, I also take up the lead in managing the system and setting up one part exclusively and another person takes up uh, the same way and trying to manage the site, if we have a site coming up, trying to set it up, uh, managing different boxes.
>
> (Anad, Developer, Company I)

Project Managers and other managers are still located atop worker hierarchies in this organizational form. In general, managerial hierarchies are stable and enduring in either type of hierarchy; it is mostly the relationships among Developers

and Team Leads that change. In a project-based hierarchy, the worker hierarchy is constructed, deconstructed, and re-constructed as projects are completed and replaced by new ones.

> We have technical leads. They would not be titled. Okay, we had like a Project Manager and Technical Lead. So Technical Lead would be a senior technical guy just advising the team. Technical Lead, because that was more on a project by project basis … but he wasn't their boss, real boss, because they worked on different projects under different bosses.
>
> (Antonio, Co-founder/COO, Company O)

Because individual developers move into and out of the Team Lead position, it is not part of anyone's title. They fulfill much of the same role as Team Leads in a dedicated hierarchy, but serve more of an advisory function. Because of this temporality, team leads are not viewed as fulfilling managerial roles in a project-based hierarchy. They were not viewed as "bosses", or supervisors, in the normal (vertical) sense of the word. Another Team Lead, Samir, bounced back and forth between designing and leading on different projects. He reiterated Antonio's conception of the Team Lead, in terms of its temporality and lesser authority.

> There's an authority structure below [the Creative Director] if there's a project, and someone has been appointed the design lead. And then they kind of are saying how stuff is going, but you know it's not like they're your boss at that point.
>
> (Samir, Developer/Team Lead, Company N)

In project-based hierarchies, the Team Lead's position is characterized by ambiguity. On the one hand, the Team Lead is given authority over a project. They may assign tasks, monitor project status, and manage the various components. On the other hand, the Team Lead is not considered the Developer's boss. Instead, they report to a Director or Project Manager. Samir's comments reveal another important point about the Team Lead. At Company N, Team Leads are "appointed" by a Director, another indicator of the hierarchy existing above the workers. The Team Leads are not peer-elected positions and they are not shared equally among workers. This was the norm among project-based hierarchies.

In a project-based hierarchy, the Team Lead generally had less decision-making authority than a Team Lead in a dedicated hierarchy. The Lead will often still make task assignments and shape project deadlines, but they are much less likely to sanction co-workers and make hiring decisions.

> I'm still like one of their peers, you know? So… I-I am… it's difficult for me to face someone … one of these people face to face … "this is what you're doing wrong." And you know, if I was an authority figure here, it would be a different thing, but I'm not, you know?
>
> (Samir, Developer/Team Lead, Company N)

In either case, they will almost never shape company strategy, design policies and procedures, decide on pay scales, and set budgeting priorities. These were reserved for dedicated managers.

Finally, in project-based hierarchies, Developers perceived less vertical mobility within the worker hierarchy. In contrast to dedicated hierarchies, the Team

Lead was not seen as a promotion, because it was a temporary assignment and many Developers could achieve Team Leads on different projects.

> *Question*: Overall, do you see a lot of mobility amongst the developers?
>
> Not really. Everyone, you know, I think has gained a lot of experience working here. But because of the size of the organization and … what direction we're moving in, there's not a lot of room for growth from a management or organizational standpoint. Your skills will grow, but there's not really any room for you to branch out and do more management. You might manage a particular project by creating a particular section of the application. But, overall, we're very targeted.
>
> (Salih, Developer, Company E)

At Company E, workers were highly specialized and focused. Unlike workers in many other software companies, they did not have opportunities to expand into managerial tasks as dedicated Team Leads or Project Managers did. With fewer layers in the worker hierarchy, the project-based hierarchy could be more flat, relative to dedicated hierarchy. The Team Lead position is less of a layer in the hierarchy. They remain more flexible from project to project and access to the Team Lead role is temporary and fluid.

In sum, there were several important differences between dedicated and project-based hierarchies. While the managerial hierarchies are similar in those two forms, the worker hierarchies (especially the Team Lead position) and mobility patterns are considerably different. In a dedicated hierarchy, a Developer acquires the Team Lead position through being promoted and subsequently gains significantly

Table 1. Comparison of Dedicated and Project-based Hierarchies

Element of Hierarchy	Dedicated Hierarchy	Project-Based Hierarchy
Temporality of Worker Hierarchy	Stable, Consistent	Fluid, Reconfigurable
Temporality of Managerial Hierarchy	Stable, Consistent	Stable, Consistent
Use of Teams	Yes	Yes
Authority of Team Leader (TL)	More authority	Less authority
Mode of Acquisition for TL	Promoted	Assigned/Replaced
Developer Has Same Supervisor From Project to Project	More common	Less common

more authority. There are three primary layers in the worker hierarchy (Project Manager, Team Lead, and Developer) with mobility among them. While this organization is stable and consistent, the project-based worker hierarchy is fluid and reconfigurable. Developers are temporarily appointed to the Team Lead position, but gain relatively less authority. Without this layer of hierarchy, Developers perceive less vertical mobility. Table 1 highlights these main differences.

Promoting the Ideology of Flatness

It is clear that neither dedicated nor project-based hierarchies are the traditional hierarchical form written about in Weber's ideal-typical bureaucracy or promoted by Taylor's scientific management. They are of a different type, or types, of hierarchy. In these new structures, some decision-making has been devolved to workers, and they have gained autonomy over certain areas of their jobs, such as limited authority over the organization of work, and task and time autonomy. They are self-managed to a degree, but at the same time, hierarchy does remain. Much high-level decision-making remains at upper levels of management and possible

career tracks remain, although not entirely, in vertical paths. Significant decision-making authority at the production-level has been devolved to workers, but authority is often further stratified among workers in the worker hierarchy.

This dual perception of hierarchy is no accident. Executive officers repeatedly discussed strategic utilization of informality, openness, and shared decision-making with workers, or developers/engineers. Luke, the CEO and founder of his company, found that this environment can increase efficiency.

> I also find it to be productive that when something comes up that is not confidential that [the engineers] can hear about it, that they can feel like, "hey, I am involved in the whole process by the way he is doing the deal." Or talking to an investor while that is exciting. I can learn from that or I can understand the whole process better when they all contribute.
>
> *Question*: How much input do they have?
>
> Lots of input. At the end of the day I make the decisions. That's the way it has to be I guess. You got to have a hierarchy at some level. You have to work. (Luke, Founder/CEO, Company H)

Luke permitted openness and solicited input in a strategic manner, but only when it does not get in the way of his control over the organization. Worker participation in decision-making did have economic benefit to the company, but he was clear about being the decision-maker. He legitimated this supposed need for hierarchy in terms of fulfilling organizational needs. John, a VP of Product Development, communicated the same point in slightly different terms.

> Most of the people on the team, except for some of the newer people, are more than comfortable with walking in here, shutting the door and talking candidly. And that's kind of the lack of chain of—if you think about it, that's kind of the lack of chain of command, the fact that people feel comfortable talking candidly … but when it comes down to it, there's a clear chain of command. I know exactly what I'm – [the CEO] tells me exactly what I'm responsible for. And then I delegate that and tell people what they're responsible for.
>
> (John, VP of Product Development, Company D)

For John, a high degree of informality and accessibility in interpersonal interactions does not mean equality and flatness in positional authority within the organization. Hierarchy still remains in his company, despite greater ability to talk frankly with individuals at different levels. For Ahmet, a CTO and founder, a flatter, more equal distribution of decision-making authority would harm the company.

> You have to be democratic at the right times. And you have to be socialist at the right times. You can't afford to be democratic all the time because it will hurt you, 'cause you can't afford to leave decision-making to, you know, everyone in the company. But certain cases, you have to make them feel, you have to enable them, you have to let them know that they're participating in the process by being democratic.
>
> (Ahmet, Founder/CTO, Company L)

For Ahmet, being a "socialist" meant being autocratic. According to him, the authority that he possessed and deployed as the autocrat was necessary to guide the

company. For Ahmet, workers did not have the capacity, or maybe the incentive, to make the appropriate decisions. However, he felt that to maximize their productivity and get them to accept his decisions, he required their consent. To do so, he fostered an ideology of democracy and flatness.

As we can see from these quotes, workers are subordinated, vis-à-vis management, through formal positions and decision-making authority. Behind an ideology of democracy, involvement, and participation, there exists a clear hierarchical structure that influences the distribution of power and authority between management and workers. Management, while allowing certain decision-making to devolve to workers, was strategic in their selection of worker participation, and targeted types of authority (high-level versus production-level) to specific positions within the organization. The evidence shows that this hierarchical distribution extends into both the managerial and work hierarchies, even if those worker hierarchies are less rigid than traditional bureaucracies.

Summary of Findings

The evidence presented here reveals many interesting things about hierarchy in software companies. First, worker hierarchies existed as components within broader organizational hierarchies. Worker hierarchies were positioned below managerial hierarchies with varying (2-4) levels. Together, they comprised a multi-tiered structure.

Second, worker hierarchies maintained similar positions across the organizations sampled here (Project Manager, Team Lead, and Developer). While all

worker hierarchies used teams and were positioned under stable managerial hierarchies, the relationships among these positions were different in dedicated and project-based hierarchies. In dedicated hierarchies, individuals were fixed, or dedicated, to their positions. The structure of the hierarchy was stable and consistent over time. Developers were promoted into the Team Lead position and gained significant authority over their subordinates. Team Leads and Developers were respectively more likely to work below the same Project Managers and Team Leads from project to project.

In a Project-Based Hierarchy, worker hierarchies were much more fluid and reconfigurable. Developers were temporarily assigned (by managers) to function as Team Lead on a given project. The assignment was not viewed as a promotion because the Developer would later be unassigned at the project completion, and would function as a Developer on other projects. In this temporary position of a authority, the Team Lead was granted some decision-making authority but was not viewed as the Developers' supervisor in the same way as a dedicated hierarchy. This resulted, in part, because Developers were more likely to work for different Teams Leads and Project Managers on different projects.

Several conclusions may be drawn from the flatness of these companies' hierarchies. First, in the general literature, flattening out explicitly and implicitly refers to the delayering or reduction of hierarchical levels. It is assumed that delayering results in the diffusion of decision-making authority (Harley 1999), and many times this is the case. However this unitary diffusion obscures the complexity

of organizational restructuring. The evidence here portrays a much more nuanced, and perhaps multidimensional, picture.

In order to tease out these several nuances, it is helpful to compare these organizational hierarchies with ideal-typical organizational bureaucracy. The traditional bureaucracy is known to have a high number of layers between top and bottom. At the bottom of the hierarchy, workers' decision-making authority in both high-level functions (i.e. resource allocation; budgeting; hiring, firing, and promotions; drafting policies and procedures; the distribution of rewards) and production-level functions (i.e. developing schedules and setting deadlines; documentation requirements; coding standards, including complexity; selecting features for a project; task assignment) is low. Furthermore, workers rarely have input on such matters. Workers also have very little individual autonomy (i.e. task and time aunotomy). However, because of the rigid pyramidal hierarchy, workers had high opportunities for vertical mobility and paths were clearly defined.

On the other hand, the number of layers in sampled software companies would be less than those in traditional bureaucracies. But like traditional bureaucracies, workers rarely had decision-making authority, or even input, in high-level decisions. Workers did have greater authority in production-level decisions, but even here such authority was distributed hierarchically among both management and workers. Production-level decisions were split between workers and management, with workers having a high level of input. As Wright et al. (1995) discuss, the ability to offer input and make decisions are not equal. Decision-making authority is a gate keeping position and a vehicle for exercising power over others within the

Table 2. Comparing Hierarchies in Traditional Bureaucracies and Software Hierarchies

Elements of Hierarchy	Traditional Bureaucracy	Software Hierarchies
Number of Layers	high	low-med
Workers make high-level decisions	low	low
Workers input on high-level decisions	low	low
Workers make production-level decisions	low	med
Workers input on production-level decisions	low	high
Workers have individual autonomy	low	high
Opportunities for and Clarity of Vertical Mobility	high	low-med

organization, while the solicitation of input guarantees no such control. Nonetheless, the degree of input and informal advice offered by Developers was not entirely trivial, in comparison to traditional bureaucracies. Managers sometimes relied on the expert advice of their Team Leads and Developers (e.g. in hiring new Developers). Furthermore, workers exercised high levels of individual autonomy, but had lesser opportunities for vertical mobility. Table 2 summarizes these differences between traditional bureaucracies and the hierarchies of software companies sampled here.

Finally, the degree of input and decision-making that was devolved to workers was not a universal or benign shift in authority. Rather, managers strategically permitted workers authority to promote the ideology of “democracy”, “participation”, or flatness. Devolving such decision-making authority to workers not only provided economic advantages of flexibility and knowledge utilization, but also obfuscated more explicit authority attributed to these positions. Supervisors and managers were able to override input from Developers, and workers were excluded from higher-level decisions entirely.

Chapter 6: Discussion

The findings in this study provide little support that hierarchies have been replaced by flat, horizontally coordinated structures. This empirical analysis has demonstrated that a more nuanced form of organizational restructuring has occurred, with varying dimensions of flatness. The optimistic idea that hierarchies have become obsolete and that they have been replaced by more flat and humane team-based structures is more a by-product of conceptual inversion (and perhaps wishful thinking!) than empirical analysis. In the words of Barley and Kunda (2001: 77), "conceptual inversion occurs when theorists formulate images of postbureaucratic organizing by contrasting traditional models of organizing with their perceived opposites." Concepts such as "network organization" have emerged and been presented as the opposite of hierarchy and bureaucracy to create sharp contrasts between past and present. These sharpened concepts have since become fads, much like Post-Fordism did in the 1980s and worker empowerment did in the 1990s (Harley 1999). In this debate (and likely in the case of other such fads), such inverted claims of flatness have not been tested (with the exception of Rajan and Wulf (2006), who empirically assessed the flattening of managerial hierarchies). The evidence presented above fills this void by constructing an empirically-grounded image of modern forms of organizational structures in software development companies.

The findings presented in this study favor the rejection of such conceptual inversions that have plagued much of organizational theorizing on "postbureaucratic" work organizations. Unlike the teams studied by Barker (1993), these software

companies did not exhibit the flat structures of entirely horizontally-networked teams. Barker's (1993: 416) ethnographic study of a small manufacturing company identified a two-tiered structure whose worker teams "had to negotiate such supervisory issues as accepting responsibility, making decisions, and setting their own ground rules for doing good work, such as deciding who was going to perform which tasks, whether or not the team needed to work overtime or on weekends, and whether to hire or fire team members." In the companies studied here, like most other organizations (Acker 2006: 446; Harley 1999), power within the company and high-level decision-making remained almost entirely at higher managerial levels. Developers, Team Leads, and to some extent, Project managers, had virtually no control over company goals, strategy, resources and budgeting, the distribution of rewards, and company policies and procedures, including those for promotion, and firing workers. Furthermore, decision-making authority remained stratified within both managerial hierarchies (Rajan and Wulf 2006) and worker hierarchies. Many companies maintained dedicated hierarchies with production-level decisions (regarding project specifications and functionalities, schedules and deadlines, coding requirements and standards, task assignments, etc.) further stratified by position (i.e. among Project Managers, Team Leads, and Developers).

However, these companies also cannot be considered traditional bureaucracies, in the ideal Weberian sense. Significant decision-making authority has been diffused throughout the worker hierarchy and down to Developers themselves (Kraft 1999). Some decision-making was shared by teams with reduced managerial supervision. Developers typically enjoyed considerable task autonomy, time

autonomy, and input on deadlines, task assignments, and the general organization of work.

In this context, then, what kind of organizational image are we left with? Like the teams studied in Ezzamel and Willmott (1998: 391), the evidence here demonstrates "how teamwork reforms and elaborates, rather than replaces or eliminates, a traditional hierarchical system of management control." Within the teams, there was a horizontal division of labor, but clear organizational hierarchies operated above and around these teams of Developers. Coordination was not through self-managed teams, peer-elected coordinators, and ad-hoc meetings (Barker 1993), but through vertically positioned Project Managers, Team Leads, and other managers (in the case of dedicated hierarchies) or Project Managers and assigned Team Leads (in project-based hierarchies). In other words, teams were *embedded* within hierarchical structures. Furthermore, the image of a networked organization is not entirely inaccurate. Rather, "the claim that organizations are suddenly 'becoming networks' and that these network are not hierarchical is overstated" (Barley and Kunda 2001: 77). This has not been an act of replacement, but an act of reform and modification. Network analysis remains a valuable means to understand these structures, as long as one recognizes that any organization may be understood as a network and "that hierarchy is a property of a network's structure, not something that a network replaces" (Barley and Kunda 2001: 78).

Barley and Kunda (2001) noted these new structures have been described in a variety of ways, such as shamrock organizations (Handy 1989), boundaryless organizations (Arthur and Rouseau 1996), and network organizations (Powell 1990),

to name a few. Such terminology makes it clear that extant concepts and language of organizational theory do not capture and reflect the new ways of organizing work and structuring organizations today. Paradoxical concepts such as "flat hierarchies" are a reflection of researchers struggling (and failing) to come to terms with the significant changes in the restructuring of work. Barley and Kunda (2001) are right to note that new concepts are needed (see also Barley 1996). Through grounded empirical analysis, I have identified and elaborated a new image of organizing work—project-based hierarchies (in contrast to dedicated hierarchies)—in an effort to better understand these changes. This terminology is meant to simultaneously signal their temporal nature in which segments of the hierarchy can be reconfigured from project to project, and their embeddedness within hierarchical forms. It is my hope that it is neither a contradiction in terms nor too vague to lose its meaning. The weight of this concept, and any such emergent concepts, however, must stand the test of time and depth across a range of organizations.

It is worth noting that hierarchies did not acquire the same form across all organizations. Rather than pressures of institutional isomorphism (DiMaggio and Powell 1983), firms developed different means of organizing work among workers. Given that companies here operated in similar external environments (including geographical, sectoral, and market locations), this finding supports other research (Mueller 1994) that suggests that the use of teams and how they are integrated into the organizational structure can be shaped by the company's internal environment. Mueller's (1994) study of automobile companies demonstrated that a company's implementation of teamworking strategies may be constrained by structural and

historical realities, such as the company's management style and company culture. My findings on the different adaptations of team implementation into dedicated versus project-based hierarchies supports such a contingency approach. Understanding the conditions under which companies adopted a dedicated or project-based hierarchy, however, was beyond the scope of this study. My data was also not suitable to analyze how such hierarchical structures change over time and the growth of the firm. These two limitations suggest rich areas for future research.

Finally, we return to Marxist theory to help us understand how the restructuring of work is mediated by the conflictual relationship between workers and managers. Based in their differing relationship to the mode of production, workers and managers have divergent interests in controlling production (Harley 1999; Ezzamel and Willmott 1998). For example, workers, or Developers at these software companies, were motivated by writing elegant code, working on the most interesting and challenging projects, working on a variety of different projects, sought control over their own work schedule, and were unconcerned with documenting their code. However, managers sought code that did the job as simply as possible ("You look for those people who like to build a monument to themselves. You keep an eye on those people."), with no additional functionalities, and was properly documented. They assigned tasks mostly based on time and budget restraints (allowing developers to select their assignments when project schedules allowed it), sometimes requiring developers to work long hours and weekends against their will. Managers controlled how many developers were allotted to specific projects and who was hired, fired, and promoted. While contemporary labor process scholars have identified a variety of

cultural techniques (Kunda 1992; Barrett 2005; Voss-Dahm 2005; Rasmussen and Johansen 2005; Marks and Lockyer 2005; Florida 2002) to exercise managerial control, these results here show that they continue to exercise power through organizational structures as well. By virtue of their position, managers at various levels were responsible for producing surplus-value and ensuring that work was done in a manner that guaranteed as great of a surplus as possible. Furthermore, to reduce conflict in the organization, management strategically included Developers in decision-making processes "because you have to be democratic at the right times." In doing so, they promoted an ideology of flatness that obfuscated the hierarchical structure and managerial control.

Chapter 7: Conclusion

Work organizations have always been important locations for the reproduction of much inequality in society (Acker 2006). However, some scholars have argued that the restructuring of formerly hierarchical organizations into flat structures of self-managing teams has alleviated much organizational inequality and produced a new structure of egalitarianism. This transformation has been understood in the context of economic gains delivered by greater flexibility, increased flow of information, and motivational qualities of worker autonomy. Culturally, workers in software development have come to expect and demand such autonomy, furthering the push toward flat, horizontally coordinated structures.

This study was meant to contribute to such debates by evaluating the nature of worker hierarchies in software companies. It has yielded several general conclusions. First, my results indicate that companies are not flat, but instead that hierarchies remain. The degree to which they have flattened out depends on the measures of hierarchy considered (layers, high-level decision-making, production-level decision-making, etc.) Secondly, I have outlined several dimensions along which flattening can occur and indicated how companies here have become more flat. Third, I have elaborated precisely how teams are embedded within organizational hierarchies through the devolution of decision-making authority into quasi-managerial positions within the worker hierarchy, and the strategic distribution of authority among those positions. Fourth, I have emphasized the need for new concepts in organizational theorizing by demonstrating both the inadequacy of existing concepts and identifying

a new form of hierarchical relations (project-based hierarchies). Fifth, because companies exhibited two distinct forms of hierarchy, I highlighted the usefulness of a contingency approach in analyzing organizational hierarchies in knowledge work. Finally, I argued that the gains in productivity achieved by transforming organizational hierarchies into flat, team-based structures, are mediated by the divergent interests between management and workers, as characterized within a Marxist framework. Hierarchy, therefore, remains a defining characteristic of software companies, and a mechanism for the reproduction of organizational and social inequality.

Appendices

Sample Questions: Programmer Questionnaire

6. Has your title changed since joining the company?
   a) What about the things you do, your functions, have they changed? Why?
   b) Have you moved up in the company? Among programmers?
   c) How has your salary changed as a result?
   d) Has there been much mobility among the programmers or not?

7. We know that some companies have many levels while other companies are very flat, can you tell us about the structure of your company?

8. Are there different titles for different programmers? What are these and what do they represent?

   a) Do these different titles mean anything? Do people with different titles perform
   different tasks? Have different responsibilities?
   b) Is this true most of the time, or just sometimes?c) Is there some overlap in the tasks performed by programmers with different titles? PROBE
   c) Is every programmer able to do the same things?
   d) What about specialization? How much are programmers specialized here?
   e) Please describe each specialty. (Languages?)
   f) Do you ever feel the work you do is routine or over-specialized? Why?

12. Let's talk about this in relation to a specific project that you're working on now.

   a) Would you briefly describe this project (or application)?
   b) Who's involved at these various stages (programmers/developers, project managers, technical leads)?:
      a)
      b)
      c)?....
   c) Others involved?
      ~Testing, Implementation, or Release groups ?
      ~others?
   d) Is this typical of other projects you have worked on?

   e) For this project, how did this process workout for you?
   f) Overall, how do you feel about this process?

g) Are there any changes you would like to see in this process? If so, what are these?

h) How did the implementation of this process change your job?

13. Let's talk a little about the stage during which code for the project is actually being written.

a) Who were/are the people involved and how were/are they organized?

b) Who was the team leader?

c) Others under the leader?

d) How did you decide on assignments? Did the programmers have a choice?

e) Do people always perform the same tasks? Or do they move around?

14. We've heard a bit about "builds" and other types of regular meetings where project members coordinate and compile work and discuss project progress....

a) Do you have these at your company? What does you company call them?

b) Do you attend these? Do you participate?

c) Does someone "run" the meeting? Who?

15. In general, how does management communicate with you?

a) Regular meetings or "builds"?

b) Emails?

c) Memos?

d) One-on-one in person meetings

18. We hear a lot about deadlines on projects.... a) How do these affect the hours that you work?

a) Do deadlines ever change or move?

b) Who creates the deadlines? Did you have any input in this decision process?

c) Are there any incentives for meeting deadlines or completing project early?

d) Do you feel that the deadlines are reasonable?

e) How do you meet deadlines?

25. How does management manage you? (Probe)

a) Is there a specific management paradigm at this firm?

b) Is this ever problematic?

c) How do you feel about this?

26. Does management do anything to increase the productivity of code writing?

Sample Questions: Project Manager/Technical Lead Questionnaire

Note: These questions are supplemental to the questions above.

12. Could you please describe your role as a technical lead or project manager within _____(company name)_____?

a) What are your specific responsibilities?

13. What is your relationship to the software programmers and developers?
    a) Supervisory/Managerial or Peer or Cooperative?

14. What is your relationship to upper (executive) management?

20. How do the programmers work with other programmers on this project?
    a) Together or independently?
    b) Where is everybody else when you're working?
    c) Are there any specific problems with this arrangement?

29. How do you manage your programmers? What is the management philosophy/paradigm of this firm?

    a) Is this the philosophy you use?
    b) Is this ever problematic?

Bibliography

Acker, Joan. 2006. "Inequality Regimes: Gender, Class, and Race in Organizations." *Gender and Society* 20:441-464.

Alcaly, Roger. 2003. *The New Economy*. New York: Farrar, Straus, and Giroux.

Andrews, Christopher, Craig Lair, and Bart Landry. 2005. "The Labor Process in Software Startups: Production on a Virtual Assembly Line?" in *Management, Labour Process and Software Development*, edited by R. Barret. New York: Routledge.

Appelbaum, Eileen, Thomas Bailey, Peter Berg, and Arne L. Kalleberg. 2000. *Manufacturing Advantage: Why High-Performance Work Systems Pay Off*. Ithaca: Cornell University Press.

Arthur, Michael and Denise Rousseau. 1996. *The Boundaryless Career: A New Employment Principle for a New Organizational Era*. New York: Oxford University Press.

Bailyn, Lotte. 1988. "Autonomy in the Industrial R&D Lab." in *Managing Professionals in Innovative Organizations*, edited by R. Katz. New York: Ballinger.

Baldry, Chris, Dora Scholarios, and Jeff Hyman. 2005. "Organizational Commitment Among Software Developers." in *Management, Labour Process and Software Development*, edited by R. Barret. New York: Routledge.

Barker, James. 1993. "Tightening the Iron Cage: Concertive Control in Self-Managing Teams." *Administrative Science Quarterly* 38:408-437.

Barley, Stephen. 1996. "Technicians in the Workplace: Ethnographic Evidence for Bringing Work into Organizational Studies." *Administrative Science Quarterly* 41:404-441.

Barley, Stephen and Gideon Kunda. 2001. "Bringing Work Back In." *Organization Science* 12:76-95.

Barrett, Rowena. 2005. "Managing the Software Development Labor Process: Direct Control, Time and Technical Autonomy." in *Management, Labour Process and Software Development*, edited by R. Barrett. New York: Routledge.

Bell, David. 2001. *An Introduction to Cybercultures*. New York: Routledge.

Bergquist, Magnus. 2003. "Open-Source Software Development as Gift Culture: Work and Identity Formation in an Internet Community." in *New Technologies at Work: People, Screens, and Social Virtuality*, edited by C. Garsten and H. Wulff. NY: Berg.

Bernstein, Ann and Peter Berger. 1998. *Business and Democracy: Cohabitation or Contradiction?* Washington, DC: Pinter.

Bloomfield, Brian. 1989. "On Speaking About Computing." *Sociology* 23:409-426.

Braverman, Harry. 1975. *Labor and Monopoly Capital: The Degradation of Work in the Twentieth Century*. New York,: Monthly Review Press.

Bray, Mark and Littler, Craig. 1988. "The Labor Process and Industrial Relations: Review of the Literature." *Labour and Industry* 1:551-587.

Burris, Beverly. 1998. "Computerization of the Workplace." *Annual Review of Sociology* 24:141-157.
Cappelli, Peter and David Neumark. 2001. "Do "High-Performance" Work Practices Improve Estalishment-level Outcomes?" *Industrial & Labor Relations Review* 54:737-775.
Caroli, Eve and John Van Reenen. 2001. "Skill-Biased Organizational Change? Evidence from a Panel of British and French Establishments." *Quarterly Journal of Economics* 116:1449-1492.
Case, Peter and Erik Pineiro. 2006. "Aesthetics, Performativity and Resistance in the Narratives of a Computer Programming Community." *Human Relations* 59:753-782.
Chandler, Alfred. 1984. "The Emergence of Managerial Capitalism." *Business and History Review* 58:473-503.
Charmaz, Kathy. 1983. "The Grounded Theory Method: An Explication and Interpretation." in *Contemporary Field Research: A Collection of Readings*, edited by R. Emerson. Prospect Heights, IL: Waveland Press.
Cloke, Ken and Joan Goldsmith. 2002. *The End of Management and the Rise of Organizational Democracy*. San Francisco: Jossey-Bass.
Coleman, Henry. 1996. "Why Employee Empowerment is Not Just a Fad." *Leadership and Organization Development Journal* 17:29-36.
Collom, Ed. 2003. "Two Classes and One Vision? Managers' and Workers' Attitudes Towards Workplace Democracy." *Work and Occupations* 30:62-96.
DiMaggio, Paul and Walter Powell. 1983. "The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizaitonal Fields." *American Sociological Review* 48:147-160.
Drucker, Peter. 1988. "The Coming of the New Organization." *Harvard Business Review*:45-53.
Dyer-Witheford, Nick. 1999. *Cyber-Marx: Cycles and Circuits of Struggle in High-Technology Capitalism*. Urbana: University of Illinois Press.
Eriksson, Tor. 2001. "The Effects of New Work Practices - Evidence from Employer-Employee Data." in *International Conference on Organizational Design, Management Styles, and Firm Performance*. University of Bergamo.
Ezzamel, Mahmoud and Hugh Willmott. 1998. "Accounting for Teamwork: A Critical Study of Group-based Systems of Organizational Control." *Administrative Science Quarterly* 43:358-396.
Florida, Richard. 2002. *The Rise of the Creative Class*. New York: Basic Books.
Gittelman, Maury, Michael Horrigan, and Mary Joyce. 1998. ""Flexible" Workplace Practices: Evidence from a Nationally Representative Survey." *Industrial & Labor Relations Review* 52:99-115.
Glaser, Barney G. and Anselm L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine Pub. Co.
Handy, Charles. 1989. *The Age of Unreason*. Boston, MA: Harvard University Press.
Harley, Bill. 1999. "The Myth of Empowerment: Work Organisation, Hierarchy, and Employee Autonomy in Contemporary Australian Workplaces." *Work, Employment and Society* 13:41-66.

Harrison, Bennett and Barry Bluestone. 1988. *The Great U-turn: Corporate Restructuring and the Polarizing of America*. New York: Basic Books.
Heckscher, Charles. 1994. "Defining the Post-Bureaucratic Type." in *The Post-Bureaucratic Organization*, edited by C. Heckscher and A. Donnelson. Thouasand Oaks, CA: Sage.
Himanen, Pekka. 2001. *The Hacker Ethic, and the Spirit of the Information Age*. New York: Random House.
Jaffee, David. 2001. *Organization Theory: Tension and Change*. Boston: McGraw Hill.
Kalleberg, Arne L., Peter V. Marsden, Jeremy Reynolds, and David Knoke. 2006. "Beyond Profit? Sectoral Differences in High-Performance Work Practices." *Work and Occupations* 33:271-302.
Kanter, Rosabeth Moss. 1989. *When Giants Learn to Dance*. New York: Simon and Schuster.
Kraft, Philip. 1999. "To Control and Inspire: US Management in the Age of Computer Information Systems and Global Production." Pp. 17-36 in *Labor and Monopoly Capital in the Late Twentieth Century*, edited by M. Wardell, Peter Meiksins, and Thomas Steiger Albany, NY: SUNY Press.
Kunda, Gideon. 1992. *Engineering Culture: Control and Commitment in a High-Tech Corporation*. Philadelphia: Temple University Press.
Lévy, Pierre. 2001. *Cyberculture*. Translated by R. Bononno. Minneapolis: University of Minnesota Press.
Levy, Steven. 1984. *Hackers: Heroes of the Computer Revolution*. Garden City, N.Y.: Anchor Press/Doubleday.
Lewis, Michael. 1999. *The New New Thing: A Silicon Valley Story*. New York: W. W. Norton.
Marchington, Mick. 1992. *Managing the Team: A Guide to Successful Employee Involvement*. Oxford, OX, UK ; Cambridge, Mass., USA: Blackwell Business.
Marks, Abigail and Cliff Lockyer. 2005. "Professional Identity in Software Work." in *Management, Labour Process and Software Development*, edited by R. Barret. New York: Routledge.
Miles, Mattew and Micahel Huberman. 1984. *Qualitative Data Analysis: A Sourcebook of New MEthods*. Beverly Hills, CA: Sage.
Mueller, Frank. 1994. "Teams Between Hierarchy and Commitment: Change Strategies and the 'Internal Environment'." *Journal of Management Studies* 31:383-403.
Negri, Antonio. 1989/2005. *The Politics of Subversion: A Manifesto for the Twenty-first Century*. Cambridge, UK ; Malden, MA: Polity Press.
Osterman, Paul. 1994. "How Common is Workplace Transformation and Who Adopts It?" *Industrial and Labor Relations Review* 47:173-188.
—. 2000. "Work Reogranization in an Era of Restructuring: Trends in Diffusion and Effects on Employee Welfare." *Industrial and Labor Relations Review* 58:179-196.
Pil, Frits and John Paul MacDuffie. 1996. "The Adoption of High-Involvement Work Practices." *Industrial Relations* 35:423-455.

Piore, Michael J. and Charles F. Sabel. 1984. *The Second Industrial Divide: Possibilities for Prosperity*. New York: Basic Books.
Powell, Walter. 1990. "Neither Market Nor Hierarchy: Network Forms of Organization." in *Research in Organizational Behavior, vol. 12,* edited by B. Staw and L. Cummings. Greenwich, CT: JAI Press.
Rajan, Raghuram and Julie Wulf. 2006. "The Flattening Firm: Evidence from Panel Data on the Changing Nature of Corporate Hierarchies." *Review of Economics and Statistics* 88:759-773.
Rasmussen, Bente, and Johansen, Birgette. 2005. "Trick or Treat? Autonomy as Control in Knowledge Work." in *Management, Labour Process and Software Development,* edited by R. Barret. London: Routledge.
Ross, Andrew. 2003. *No-Collar: The Humane Workplace and its Hidden Costs*. New York, NY: Basic Books.
Sproull, Lee, Sarah Kiesler, and David Zubrow. 1984. "Encountering an Alien Culture." *Journal of Social Issues* 40:31-48.
Stewart, Thomas A. 1997. *Intellectual Capital: The New Wealth of Organizations*. New York: Doubleday / Currency.
Stohl, Cynthia and George Cheney. 2001. "Participatory Processes/Paradoxical Practices: Communication and the Dilemmas of Organizational Democracy." *Management Communication Quarterly* 14:349-407.
Turkle, Sherry. 1995. *Life on the Screen: Identity in the Age of the Internet*. New York: Simon & Schuster.
Turner, Fred. 2006. "How Digital Technology Found Utopian Ideology: Lessons From the First Hacker's Conference." Pp. xvii, 323 p. in *Critical Cyberculture Studies,* edited by D. Silver and A. Massanari. New York: New York University Press.
Voss-Dahm, Dorothea. 2005. "Coming and Going at Will? Working Time Organization in German IT Companies." in *Management, Labour Process and Software Development,* edited by R. Barrett. New York: Routledge.
Weiss, Robert Stuart. 1994. *Learning From Strangers: The Art and Method of Qualitative Interview Studies*. New York: Free Press.
Wellman, Barry. 1997. "An Electronic Group is Virtually a Social Network." in *Culture of the Internet,* edited by S. Kiesler. Mahway, NJ: Lawrence Erlbaum.
Wellman, Barry and Milena Gulia. 1999. "Virtual Communities as Communities: Net Surfers Don't Ride Alone." Pp. x, 323 p. in *Communities in Cyberspace,* edited by M. A. Smith and P. Kollock. New York: Routledge.
Whyte, William Hollingsworth. 1956. *The Organization Man*. New York,: Simon and Schuster.
Woodfield, Ruth. 2000. *Women, Work and Computing*. Cambridge, UK ; New York, NY, USA: Cambridge University Press.
Wright, Erik Olin, Janeen Baxter, and Gunn Elisabeth Birkelund. 1995. "The Gender Gap in Workplace Authority." *American Sociological Review* 60:407-435.
Zmud, Robert. 1982. "Diffusion of Modern Software Practices: Influence of Centralization and Formalization." *Management Science* 28:1421-1431.