THE LABOR PROCESS OF SOFTWARE DEVELOPMENT IN TURKEY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF SOCIAL SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HÜSEYİNCAN ERYILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE PROGRAM OF MEDIA AND CULTURAL STUDIES

SEPTEMBER 2015

Approval of the Graduate School of Social Sciences

_____

Prof. Dr. Meliha Altunışık
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Necmi Erdoğan
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Instr. Dr. Barış Çakmur
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Necmi Erdoğan	(METU, ADM)	_____
Instr. Dr. Barış Çakmur	(METU, ADM)	_____
Prof. Dr. Gamze Yücesan-Özdemir	(Ankara Uni, JRM)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name :

Signature :

ABSTRACT

THE LABOR PROCESS OF SOFTWARE DEVELOPMENT IN TURKEY

Eryılmaz, Hüseyincan
M.Sc., Media and Cultural Studies Master Program
Supervisor: Instr. Dr. Barış Çakmur

September 2015, 181 pages

This study examines the labor process of software development which is situated in a central position within the information and communication technologies industry. Based on the fact that production of software applications has not received much academic interest, software production in Turkey is tried to be illuminated in the light of interviews made with workers in software sector. In this respect, a general framework is tried to be presented about how and in which relations through software is developed; and how those relations are experienced and perceived by software workers in Turkey. Technical organization of software development, relations of control at workplace, role of skills in production process and job insecurity experienced by workers in the sense of uncertain future are discussed. In this regard, the field research of the study was conducted through semi-structured in-depth interviews with 21 software developers who are working in different private companies in Ankara and İstanbul.

Keywords: Enformation and Communication Technologies, Software, Software Development, Skill, Job Insecurity

ÖZ

YAZILIM GELİŞTİRMENİN TÜRKİYE'DEKİ EMEK SÜRECİ

Eryılmaz, Hüseyincan
Master., Medya ve Kültürel Çalışmalar Master Programı
Danışman: Öğr. Gr. Dr. Barış Çakmur

Eylül 2015, 181 sayfa

Bu çalışma, enformasyon ve iletişim teknolojileri endüstrisi içerisinde merkezi bir noktada konumlanan yazılım geliştirmenin emek sürecini incelemiştir. Hayatın her alanında kullanılan yazılım uygulamalarının üretiminin çok fazla akademik ilgi görmemiş olması noktasından hareketle, Türkiye'de gerçekleştirilen yazılım üretimi, sektörün çalışanları ile yapılan görüşmeler ışığında aydınlatılmaya çalışılmıştır. Bu bağlamda, yazılım uygulamalarının hangi üretim ilişkileri içerisinde, nasıl geliştirildiği ve bu ilişkilerin yazılım işçileri tarafından nasıl deneyimlendiği ve değerlendirildiğine dair genel bir çerçevenin sunulmasına gayret edilmiştir. Yazılım üretiminin teknik organizasyonu, iş yerindeki denetim ilişkileri, vasıfların üretim içerisindeki rolü ve geleceklerinin belirsiz olması bağlamında yazılım çalışanlarının deneyimlediği iş güvencesizliği konuları ele alınmıştır. Bu doğrultuda, Ankara ve İstanbul'da farklı özel şirketlerde çalışan 21 yazılım geliştiricisi ile yarı-yapılandırılmış derinlemesine mülakatlar gerçekleştirilmiştir.

Anahtar Kelimeler: Enformasyon ve İletişim Teknolojileri, Yazılım, Yazılım Geliştirme, Vasıf, İş Güvencesizliği

ACKNOWLEDGEMENTS

Firstly, I would like to thank to my interviewees who accepted my request of interview and sincerely shared their experiences despite their lack of time. This thesis would not be meaningful without their valuable sharings and experiences which enable me to make this research. I want to express my deepest gratitude to my advisor, Barış Çakmur who has supported me patiently throughout this study. I consider myself lucky for having the experience of working with him. I am also grateful to my examining comitee members Gamze Yücesan-Özdemir and Necmi Erdoğan for their precious comments and advices.

If my friends and family were not there for me, this thesis would have never be completed. There is no word which I can use here to express my gratefulness to Eda, Demet and Alişan for helping me by editing the drafts of this thesis without any complaints. Their friendship and moral motivation gave me the power for final blow to implement this study. I am also grateful for his care and support to Necip, the best flatmate one can ever be in every but every respect. I would like to thank to Helin, Kadir, Sarper, Ceylin, Ceren and Yiğitalp. In most desperate times I had been through, they were all with me with their endless support, joy and love. I am also grateful to Gökhan, Ozan, Recep and Ceyda for having me study in their cosy workplaces which we had lovely time together. Our hours of companionship which were full of great stories and inspirational conversations were refreshing for me. I would also thank to Ali, Sinan, Burcu, Damla and Mert for patiently listening my never ending concerns and whinings related to the thesis. Their encouraging words were big help for me to hold on this study. I also want to thank Elif who proofread my thesis in a very limited time.

Lastly, I owe my deepest gratitude to my family for their indispensable and endless support. Without their love and care, I would not accomplish this study.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

BMO	Bilgisayar Mühendisleri Odası (Chamber of Computer Engineer)
CASE [Tools]	Computer Aided Software Engineering [Tools]
COBOL	Common Business Oriented Language (Programming Language)
FORTRAN	Formula Translating System (Programming Language)
ICT	Information and Communication Technology or Technologies
IDE	Integrated [Software] Development Enviroment
SO	Stack Overflow (Question and Answer Website)
OOD	Object Oriented Design (Programming Paradigm)

CHAPTER I

INTRODUCTION

1.1 Objective of the Study

Since 1970s, information and communication technologies [ICT] have gained significance mainly upon the society and general production organization. There is a consensus among scholars that due to these technologies, series of transformation whether it represents rupture or continuity from past have taken place in general organization of production, thereby in the society. Information based activities have created new occupations and permeated into agricultural and industrial manufacture, as well as every corner of people's social activities.

Software applications, hence software development are crucial in drastic proliferation of these technologies because, modern computers' becoming so vital in our lives has been achieved remarkably with the advancements of software applications. Therefore, given the centrality of information based products as a commodity and characteristics of the contemporary economy, software development as an occupation is deemed to be one of the most important aspects of this era. Software developers as part of 'knowledge workers' are even named as the aristocrats of the labour market (Castells, 1996).

According to the dominant discourse, software developers are intellectual and creative; they choose the work they engage in and they are self-administrative (İliç, 2011). Complex processes such as acquiring information, evaluation, organization and analysis of it; and providing information when it is needed are the main

constituents of software developers' labor. However, in these approaches which draw a very positive scene, capitalist production relations and domination relations are ignored. The class structure and the relations of production software development is engaged in are overlooked. In this regard, as Yücesan-Özdemir (2002, 435) points out, a theoretical approach that analyzes the capitalist labor process as well as dynamics that shapes it have hardly been put forward.

It should be stated that, in labor process of software development, relations of production and control of capital over the labor have some peculiar characteristics compared to the other fields of production. Even so, reproduction, accumulation and the control of the capital have also become integrated into developed capitalist control mechanisms in the software development sector, too. Therefore, even though it can be argued that software developers have still certain degree of autonomy over their own labor; this does not change the fact that as wage laborers, they have to sell their labor as commodity to capitalists, so they are subjected to the capitalist relations of production.

Recent studies made on educated and skilled workers in Turkey show that, 'white collar' and 'knowledge workers', except few who are employed in elite positions, remarkably lose their job security and privileges in terms of working conditions once provided by skills and qualifications they have (see Bora et al, 2011; Sağıroğlu, 2013; Yücesan-Özdemir, 2014). It seems that threat of job insecurity as well as harsh working conditions have started to be experienced among a wider spectrum of skilled workers along a line from upper middle class to lower middle class.

Except the rare, thereby valuable studies, which focus on skilled workers within ICT sector[1], specific studies on how and through which relations software development process in Turkey is attained are hardly found. There are some researches done by scholars, chambers or associations about software development; however majority of them focus generally on the brightest side of the picture, or more truly to the appearing image. Most of the time, social relations of production are excluded out of the picture. This means that not sufficient attention is given to what happens really at the production point, more precisely to the production relations of the software development process.

The main questions of this study are therefore to what kind of specific technical organization structure which software development in Turkey show, how workers control over production is affected from that specific setting of production organization, what kind of forms which relations of control over production process take at workplace, how managers apply strategies of control and software developers react to these strategies and how software workers perceive relations of production in regards of job insecurity and take actions accordingly.

This study develops three main arguments in regards of these research questions. First of all, it is asserted that incessant transformation of technical organization of software development has shown tendencies towards standardization and automation of production. As a result, control of production process has passed to some extent to the hands of management with the help of means of software production. Consequently, effectiveness and control of software developers as direct producers over production process are getting diminished along with their value of working skills and qualifications. Accordingly, it seems that workers are losing their privileges in the sense of working conditions. Secondly, it is proposed that as well as implementation of the direct control strategies, relative autonomy is bestowed to

[1] To see these studies, look at Kodalak (2007), İliç (2011) and Armağan (2012)

software developers at workplace as to increase and secure the rate of profitability. It means that labor process of software development and capitalist control at workplace, including managerial strategies based on consent and collaboration of workers have been also formed around imperatives of capitalist accumulation. Finally, destructive effects of flexible labor regime along with short-term employment relations keep pressure of uncertain future and job insecurity on software workers perpetuated. As a result, workers tend to accept exploitation of their own and to participate reproduction of its conditions by deploying individual and market-oriented strategies to stay competitive and actively employable in labor market.

In this respect, in this study, I intend to focus on labor process of software development which is situated in a central position within the information and communication technologies industry. The main aim of this study is to obtain an idea about in which relations through software is developed, how the work is organized and software developers managed and what this means in terms of their commitment to their work and employment. The purpose is to find certain uniformities and peculiarities in labor process of software development regarding to capitalist production relations as well as to draw a general overview about the software developers' working conditions in Turkey.

1.2 Theoretical Framework

General theoretical framework is needed to put our study's stance in a comprehensible basis. Therefore, in this part, historical settings in which proliferation of ICT has been occured is going to be briefly handled first. Afterwards, Labour Process Theory which main arguments of this study is based on will be discussed.

Beginning from 70's, numerous conceptual perspectives have been propounded and deployed to comprehend general characteristics of this era. However, we are not capable of to examine all notions developed for to delineate the major tenets of change with all details. Therefore, separation can be made and followed, at least for practical reasons between those who proclaim a new sort of society has emerged from the old, as materialized in concepts like 'information society', 'information revolution', 'knowledge society' and those who place emphasis on continuities. In this part of study, I want to briefly concentrate some of notions and approaches from these two interlinked camps in order to position our study's stance in this regard.

In 70's, innovations on information and communication technologies [ICT] have led to the appearance of different conceptions centred on technology trying to make sense of what we were living through. Main idea of those is focused upon the burgeoning effectiveness of new technologies, and diffusiveness of them into social world. It is simply asserted in these approaches that society is going to be reconstituted inevitably into a new form due to the technological innovations whose impacts are so profound. According to those, changes are the signal the coming of a new society, namely of an 'information society' or of an 'information revolution'.

From a perspective of those who take technological advance to the center of their study, extensity of developments on ICT and peculiar characteristics of those should be the evidence of new type of society, as how steam power, engine or electricity were the characteristic of industrial society (Landes, 1969; Naisbitt, 1984 cited in Webster, 2002). Computers, information, internet, thus, are supposed to be the usher of this new era. Proponents of information or post-industrial society notion tend to emphasize effects of informational and communicational technological advancements on economy and structure of general organization of production.

For those who focus effects of ICT on economy, once the greater part of economic activities like agriculture or industrial manufacture are taken up by the information activities. As a result, criticality of information based activities on economy has been remarkably increased. However, the claim that the information based activities predominate the economy is ambiguous one, since information based activities have permeated into agricultural and industrial manufacture, rather than dominated or made redundant them at all. That is to say, information activities have been incorporated into these fields and led to the transformation of them. In addition, relations between different economical activities are so intertwined and complex that makes separation of them from each other impractical. Therefore, reaching a conclusion that enhancement of information based activities within economy refers historical rupture from past eventually precludes other relations involved in this transformation.

Another approach within information society concept is emerged from ICT's, thereby information activities' effects on occupational structure of work. Salient indicator of information revolution, according to this logic, is the structural change of employment in information based works and consequently in service sector. It is stated that, information work offers more employment potential than agricultural and industrial manufacture can ever do because of the material limitations latter two have. What implied is that, information as a raw material of non-manual work contrary to agriculture and industry work, has made it possible the substantial growth in economy and increase in employment. This idea is based upon the distinctive characteristic of knowledge as raw material. Because, contrary to other inputs of production, knowledge is an inconsumable good. Therefore, an economy based on it has promising potential in the sense of growth in economy and employment. According to Bell (1973), we have entered to a new era, characterised throughout by a augmented presence and significance of information. He proposed that in post-industrial society, qualitative and quantitative shift brought into being

by information and communication technologies are evident (Bell, 1973). One can easily observe this shift by checking out the structural change in general organization of production. That is to say, in his view, the type of work that is most common becomes a defining feature of particular societies. Thus, his views suggest that while in pre-industrial societies agricultural labour is pretty well everywhere, and factory work is the benchmark in industrial societies, in post-industrial societies it is service employment based on information which predominates (Webster, 2002).

Increase in the part played by knowledge and information in social, economic and political relations is undeniable. But, interpretation of these as signalling a new type of society, is doubtful one (Webster, 2002). In this account, because, interrelations between agricultural, industrial and informational activities seem to be avoided. It is known that infrastructure of information and technology based economic activities are directly dependent on industrial outputs like energy, or required alloys composed of various minerals as well as the low skilled labor. Therefore, without considering agriculture and industry work and prevailing production relations in those within totality, the claim that information based activities are straight-forwardly liable from alleged change in economy and employment might be misleading.

In these approaches briefly mentioned above, such presupposition is dominant that technology comes from outside of society as a transforming element without any contact with it in its development and it impacts on society (Webster, 2002). However, it is hard to say from this study's point of view that technology is detached from the totality of the social relations, thereby, determinative own by own. On the contrary, technology is an essential and intrinsic part of complex social relations.

Within notions claiming that decisive break took place from industrial production organization to a new one; flexible specialisation conceptualization has an important position which needs to be mentioned. According to this approach, dominant paradigm of production organization has been changed from rigid structure to the more flexible one. Centring on work, Piore and Sabel (1984) in their study, claims that during the era of fordism, when mass production predominated, dynamics that shape the production organization were originated from tayloristic principles of management which includes rigid time and motion arrangements, hierarchical supervision and direct control techniques. In addition, in this era, standardised products were demanding specialisation of machinery and specialisation of labor, however, characterised by low levels of skill. That means what is focused on production, hence give a way to its shape was the pool of low skilled labor. However, decentralisation of large organizations, growth in subcontracting and advancements on ICT have led to radical break from this type of work organization. Vertical disintegration of work organization towards horizontal and flexible one have transformed market dynamics into a more volatile state. Thus, criticality of high skilled labor has been increased since they are more reliable and adaptable to respond changing demands of market. That is to say, shift has been actualized from the fordism which demands repetitious and low-skilled labour to the 'post-fordism' which increase the skills of employees and allow greater variety in the production of goods.

It is implied in this approach that it has been experienced a transition in all industries from mass production based vertical organization to a flexible horizontal one. Thus, according to proponents of flexible specialization notion, workers have become flexible specialists in a favourable sense, since flexible work organization has led to constant reskilling of workers (Piero and Sabel, 1984). However, it can be proposed that even reskilling is a valid phenomena for some part of workers, ongoing deskilling process seems to be missing in this approach. That is to say,

while there may be a core of skilled employees, much more vulnerable people working part-time, casually or on short-term contracts can still be identified (Gordon et al., 1983). To conclude, it can be claimed that flexible specialization approach has tendency to emphasize changes over production organization while it avoids, therefore, has difficulties to explain continuities of capitalism which are evident like domination of market criteria, wage labour, corporate organization and commodity production, establishing links with past (Webster, 2002, 96).

The same changes we have discussed till now are also be associated with and taken consideration into a set of wider shifts in what theorists of the French Regulationist School have termed the regime of capitalist accumulation (Aglietta, 1979). Accumulation regime notion, contrary to ones we briefly mentioned above, offers more holistic explanation of social relations which attempt to grasp the overall character of particular periods including effects of technology.

Thinkers of regulationist school was interested with the question: how does capitalism manage to continue in spite of all sources of tension between workers and capitalists. In this sense, they acknowledges both the changes happened on mechanisms that secure accumulation process and continuities on dynamics immanent to capitalist mode of production. Therefore, they seek to analyze mechanisms in which instabilities are handled such that continuity can be achieved during the change (Webster, 2002, 64).

As put it by Boyer (1990), each regimes of accumulation show a peculiar combination of relations through which commodity outputs are secured, the economic surplus appropriated and new investments made back into the sphere of production. That means essential characteristics of capitalist mode of production such as commodity production and appropriation of surplus value remain as cross-cutting elements of all different regimes of accumulation, while ways of securing

and obscuring accumulation process may change over time according to the conditions.

State as an intermediary between capital and labor has tried to regulate relations between two through collective bargaining and provisions of various legal, political and social rights to the citizens of national state (Jessop, 2003, 55-72). It might be briefly said that in a period from World War II till mid 70's, keynesian welfare states and fordist production organization aimed to secure full employment in a relatively closed national economy. This era was denoted as fordist accumulation regime of capitalism.

In 70's, ongoing crises of recession, unemployment and bankruptcies were indicators of an end to the fordist accumulation regime, according to proponents of regulationist school thinkers (Webster, 2002, 64). Targets of keynesian regimes such as full employment, economic growth had already been undermined by the crises and global circulation of goods, services as well as the rising conflicts between capital and labor. Fordism and keynesianism because of their rigidity is thought to be remained incapable to solve these intrinsic crises of capitalism.

To overcome crises inherent to capitalism, the Fordist accumulation regime was dismantled together with the social rights systems enhanced by the welfare states. Mode of regulation, which is based on keynesian nation state and its full authority, has also gradually lost its regulatory and intermediary functions. The suggestion is that the Fordist regime of accumulation which lasted from 1945 until the mid-1970s became unsustainable, it gave a way to a flexible regime which will restore and secure the health of capitalist enterprise (Webster, 2002, 64). This paradigm shift is accepted as the primary origin of the transformation has been brought into being.

After 1970's consequently, a new regime of accumulation, social and political readjustment "coupled with quite different system of political and social regulations has emerged" (Harvey, 1989, 145). It might be briefly said that hegemonic regime of fordist accumulation with its specific mode of regulation has been giving way to the regime of flexible accumulation.

It is claimed that unlike mass production activities which are typically rather rigid in structure, the new forms of production has emerged which are generally identified by an ability to change process and product organization great rapidity which is frequently enhanced by the use of ICT technologies (Harvey, 1989). Consequently, independent small scale firms that are specialized in a specific domain started to be preferred for the execution of the jobs which were used to be done within the internal departments of a big scale firms. As a result, a production environment is shaped in a way that many firms specialized in certain sub-sectors are hired by other firms to make a job executed (A.J. Scott, 1988: 11).

Within fordist accumulation regime, keynesian welfare states were playing a role to sustain reproduction of labor. However, crises have led to the disintegration of keynesian policies. Hegemony of capitalists over workers in this regard has lost its operability. Therefore, a need of new settings which will sustain reproduction of labor has appeared. In this sense, relations of production between workers and employers have been restructured. Through changes in production organization, redesign of labor market relations has brought advantageous position to the employers, stated by Harvey (1989, 150) as:

> [T]he labor market has undergone a radical restructuring. Faced with strong market volatility, heightened competition, and narrowing profit margins, employers have taken advantage of weakened union power and the pools of surplus (unemployed and underemployed) laborers to push for much more flexible work regimes and labor contracts. (Harvey, 1989, 150)

Flexible accumulation regime has led to the "the breaking of collective bargaining, and the stratification of labor force into a restricted upper level of highly skilled workers and a vast lower level of atomized and flexibilized individuals kept on low wages and in precarious jobs" (Neilson and Rossiter, 2008, 57). Fragmentation and segmentation of labor, parallel to the deepening horizontal alignment of production organization has been expanded.

That is to say, rising flexibility in the organization of production has corresponded to the rising flexibility in labor markets. Burden of market conditions which companies must adapt is assigned to workers through insecure and precarious employment relations. Therefore, outcomes of flexibility for employers and workers are completely different. 'What capital specifies as flexibility', as Çerkezoğlu and Göztepe argue (2010, 68) is started to be lived 'as tension by the labor'.

So, debate of the transformation is also required, referring to its outcomes for the workers. The foremost concept regarding the transformed nature of capitalism since 70's has been the precarization derived from the term "precarite" used by scholars in order to describe the temporary or seasonal workers (Standing, 2011, 9). Precariousness has been becoming the general term of newly emerging, flexible forms of employment under flexible accumulation regime.

There are innumerous approaches and definitions towards precariousness. However, there can also be mentioned of such a consensus among majority of scholars on precariousness as emanated from specific employment relations. Such concept might be helpful to specify link between redesigned employment relations and flexible organization of production. And it is possible from these discussions to derive general characteristics of precariousness employment. Rodgers and Rodgers

(1989, 3) for example, define four dimensional process of precarization of employment relations in their study as:

> (1) the degree of certainty of continuing work - precarious jobs are those with a short time horizon, or for which the risk of job loss is high. Irregular work should be included here too, insofar as there is uncertainty as to it continuing availability.
> (2) Aspect of control over work – work is more insecure the less the worker (the individual or collectively) controls working conditions, wages and pace of the work.
> (3) Protection – that is to what extent workers are protected either by law, or through collective organization, or through customary practice – protected against, say, discrimination, unfair dismissal or unacceptable working practices, but in the sense of social protection, notably to access social security benefits.
> (4) An ambiguous respect is income – low income jobs may be regarded as precarious if they are associated with poverty and insecure social insertion. (Rodgers and Rodgers, 1989, 3)

In this study however, precariousness will take place mostly in relation to the aspects of control over work, job insecurity and ambiguous future. Our focus will be to its role as a part of domination of employers on workers, under the circumstances that based on the creation of a generalized and perpetuated state of insecurity.

The utilization of precariousness as a concept is mostly identified with low-skilled and low paid jobs, however, the recent studies has shown that high-skilled and well paid works today also have precarious conditions (Magdoff, 2004). These high-skilled workers, like programmers, might have possessed individual value on the labor market, however, as it is also the case in Turkey, they do not yet have a collective force or a subjectivity regarding social rights. That means they might make above-standard wages but they have no protective rights to save them from unemployment in future or becoming redundant with aging. Of course, even though workers from different occupational positions have in common flexibilized and de-formalized labor relations, they occupy very different positions in the production

process (Candeias, 2005). In this sense, definition of Papadapoulos, Stephenson and Tsianos (2008, 250-251) highlighting what is common in precariousness for all workers is illuminatory:

> These experiences [of precarious workers] vary immensely, but they are all permeated by a pervasive social conflict: it is a conflict between high productivity and low protection, or else intensive creativity and deep vulnerability. (Papadapoulos et al., 2008, 250-251)

As indicated above, precarious employment relation with its effects on relations of production isn't a peculiar problem of certain industries or workers, rather it brings specific characteristics to all types of employments in terms of job insecurity and vulnerability. It suggests not necessarily "absolute poverty, deprivation and isolation in the workplace, but a relative disadvantage in these three dimensions."(Candeias, 2009, 8).

The labor process theory is one of the central focal point in analysis of capitalism, because it is the starting point of not only the production of surplus value but also the production relations. What makes critical the labor process of capitalist production in our account is its emphasis to the transformation occurs in accordance with the accumulation of capital; which is thought to manifest itself as continuous changes in the labor processes of each branch of industry and as a redistribution of labor among occupations and industries (Foster, 1998, xiii). Therefore, labor process theory can provide us the conceptual framework which is needed to link production relations and labor regime with wider shifts occured in capitalist accumulation regime of contemporary era.

In mid 70's, the theory of labor process has influenced many researchers who are interested in critical perspectives on work. Specifically seminal contribution of Braverman, Labor and Monopoly Capital, has reinvigorated the study of the work

place and the labor process, by reasserting work organization based on relations of production (Littler and Salaman, 1982). By insisting on the connection between two, it has promoted class theory and work analysis. Following him, numerous critical debates have been established. These debates have mainly focused upon the technical and social relations that are produced and reproduced within working place and to explain the debates on skills, control, technology and politics which are already embedded in labor processes (Özdemir&Yücesan-Özdemir, 2008, 22).

Based on reading of Marx, Braverman (1974, 35-36) begins to his work by reinstating distinction between labor and labor power. His point of departure for analysis of the unity of the labour process and capital accumulation is based on classical account of the exchange between capital and wage labour:

> The worker enters into the employment agreement because social conditions leave him or her no other way to gain a livelihood. The employer, on the other hand, is the possessor of a unit of capital which he is endeavoring to enlarge, and in order to do so he converts part of it into wages. Thus is set in motion the labor process, which, while it is in general a process for creating useful values, has now also become specifically a process for the expansion of capital, the creation of a profit. From this point on, it becomes foolhardy to view the labor process purely from a technical standpoint, as a mere mode of labor. It has become in addition a process of accumulation of capital. And, moreover, it is the latter aspect which dominates in the mind and activities of the capitalist, into whose hands the control over the labor process has passed. In everything that follows, therefore, we shall be considering the manner in which the labor process is dominated and shaped by the accumulation of capital. (Braverman, 1974, 35-36)

Labor as a commodity which can be bought, sold, used and dispensed with has different characteristic than other commodities. Because, the commodity which the worker sells is not a fixed amount of labor embodied in a completed product, but ‘labour power’; i.e. the capacity to work (Littler and Salaman, 1982).

Thus, there can be mentioned of a central indeterminacy at the translation of labor power into labor. To accomplish this translation, employers have to build structures of control over labour in order to reduce or eliminate uncertainty, while workers may respond resistance or consent to those attempts.

According to Braverman, antagonistic interests of labor and capital make control over production process obligatory for capitalists. In his study, it is proposed that relation between worker and capitalist is a contested one. Because, in the sense of exchange value, relations between capital and labor sum always is zero sum. That is to say, capitalist's gain is always equal with worker's expense or vice versa. Therefore, control over labour has been central and fundamental problem of management, and it represents itself in history as progressive alienation of the process of production from the worker (Braverman, 1974, 68).

In account of Braverman, structural organization of control of the labour process are matched with imperatives of capital accumulation. Those very imperatives, therefore, are also the dynamics give a way to the incessant transformation and tendential degradation of labour in capitalist societies (Elger, 1979). Managerial problem of control, thus, inevitably leads to "the alienation of the labor process from the laborer, that is, to the separation of manual and mental labor, or more precisely, to the separation of conception and execution parts of production" (Burawoy, 1978, 248).

The division of labor brought by scientific management characterize the separation of conception from execution. Through means of production, skill and knowledge are extracted from direct producers and placed into the hands of management. Thus, the tendencies of the labor process under the guiding principle of managerial control are towards the deskilling and fragmentation of work on one hand and the creation of an apparatus of "conception" on the other.

However, successors of labor process theory have criticised unilateral and objectivist description of dynamics of labor process of Braverman. Central criticism, crudely put, has been that depiction of a "single, overall trend- an imperative of control of the labor process" which ignores the existency and complexity of worker resistance (Littler and Salaman, 1982, 256). Braverman's analysis which is found 'exclusively from the side of the object' may prevent in this respect to see that workers are not totally ineffectual against capitalists' implementations on labor process like the taylorist practices, hence restructuring of labor process (Burawoy, 1985, 48).

Burawoy has also opposed to the Braverman's presupposition of irreconcilable interests of labor and capital as well as the constant deskilling process which is stated as inherent to capitalist mode of production. According to him, such presuppositions embody the danger of ignoring the ideological terrain where interests might be organized since act of production under capitalist mode of production, doesn't only provide exchange value but also use value (Burawoy, 1985, 26-35). As proposed by him, in the case of exchange value, relations between capital and labor may be zero sum, but "in terms of use value, it is non zero sum" (Burawoy, 1978, 256). This situation implies a possibility of conditions where the interests of capital and labor are coordinated.

For these reasons, to Burawoy, separation of conception and execution is a contingent process rather than necessity. Therefore, it can't be taken as a verbalisation of capitalist logic or law of motion determinative on labor process. In its stead, Burawoy (1985, 32) proposes that labor process and control strategies of management within should be evaluated in the sense of obscuring and securing surplus value rather than reducing or eliminating uncertainty at the translation of labor power into labor. That is to say, capitalist control is mainly operated as to

secure surplus value as well as to keep it hidden. Such approach would be well-directed according to him because, it involves not only objective elements of capital but also political and ideological aspects of work which seem to be missing in account of Braverman (Burawoy, 1985).

In that sense, Burawoy (1979, 15) develops his concept of labor process, which has two inseparable components, namely relations of production and relations in production. 'Relations of production' refers to the relations between those who produce the surplus and those who expropriate it. It defines a particular way of distributing and appropri-ating labor time and its product. 'Relations in production', on the other hand, are the relations of the shop floor into which workers enter, both with one another and with management. The transformation of nature as defined by the capitalist labor process, that is, by the relations in production, reproduces the relations of production and at the same time conceals the essence of those relations of production. In this study, even relations in production and relations of production are reciprocal aspects of labor process, main focus will be on "relations in production" of software development in Turkey.

1.3 The Research Method

In order to answer the research question(s) I have put forward for this thesis, the field study has been conducted through semi-structured in-depth interview method on the basis of snowball sampling with 21 software developers. In the beginning, I conducted a 40-question survey to determine the sampling. Through the sampling, I intended to see the distribution according to age, sex, education and geographical location. However, the survey could not attract attention even though I asked blogs widely used by software developers to be posted online. As the survey could reach only 14 people, I had to drop the survey and continue with the in-depth interviews. I

did random sampling; however I sought equilibrium among respondents in terms of age, city, and the organizational structure of the companies.

At the beginning of the interviews, personal information like age, educational formation, city of residence; social, financial and family background of respondents were taken. I started the interviews by asking workers' individual histories with software development. Then, professional career, for how long and in which companies they had been working at this sector were commonly discussed. Afterwards, I generally tried to direct the conversation to the details like their thoughts about instruments of software development, production organization at their companies, their relationship with their management, future prospects and skills adjustment strategies they employed. Workers' experience over job applications and interviews were also questioned. Even though I could not find a chance to discuss deeply in this study, I asked questions in relation to personal thoughts of software developers about the impacts of software applications on other fields of production in which those applications were implemented.

I determined basic themes and open-ended questions for interviews. However, I tried to avoid using a pre-determined question set. Those were only used when the respondents drifted away from the topic. As long as the interviewees stayed within the limit of the study's scope, I preferred listening to their reflections and experiences without interrupting. I paid attention not to interfere in their narratives. Almost all of my interviewees were helpful during the interviews, and they tried to answer all my questions straightforwardly. The only exception to this was the questions about the wages and company related information which they were employed in, and some of them openly stated that they would prefer not to answer such questions while some answered hesitantly.

All of my respondents are employed in private companies. Since working as a software developer in the public sector has significant differences compared to the private sector, sampling is limited to the private companies.

Sampling is also limited to the software developers who work in the technical division of production. Among the respondents, there were team leaders as well as senior developers; however, work content of these developers was mostly related to the technical organization of production. The reason behind this delimitation is the differences I have observed in terms of working conditions and experiences of the software developers in managerial positions.

Throughout the fieldwork, I had interviews with 21 respondents, 5 of whom were female and 16 of whom were male software developers. As far as the residency was concerned, the number of respondents residing in Ankara was 12; whereas the number of those who reside in Istanbul was 9.

To understand the experience of working conditions, age is a critical parameter since it is highly determinative on the experience level of software developers. I tried to match my sampling with distribution of workers in software development sector in terms of age. Based upon the information from surveys and researches, I preferred to give priority to younger developers. Therefore, 11 of the respondents were chosen under 30 and 10 of them were above 30 years old.

Respondents also vary in educational background; numbers can be given as 10, 7 and 4 who are graduated from computer engineering, different engineering departments and faculty of basic and applied sciences, respectively. 12 of the respondents were METU (Middle East Technical University) graduates. The others graduated from metropolitan universities in Ankara and Istanbul. I am aware that the range of universities is limited to these metropolitan universities. However, I

had problems in reaching software developers from Anatolian universities, that is, universities outside Istanbul, Ankara and İzmir.

It is known that university education is a significant indicator determining job opportunies, working conditions and average incomes. As asserted by recent studies, (Bora et al., 2011; Sağıroğlu, 2013), it is observed that metropolitan universities provide advantages to skilled workers in the sense of finding a job, better salaries and promotion opportunities in the private sector. Therefore, assumption can be made that software developers who are graduated from these universities have better and more secure working conditions. In this respect, our field study whose respondents mostly graduated from metropolitan universities may not cover working conditions of all software developers, specifically the ones who graduated from anatolian universities. This gap should be taken into consideration by readers.

Working conditions and organizational structure of those I have interviewed also differ from each other. While 3 of them work in their own startups, 1 of them is freelance worker. Other developers are contracted workers. In terms of company scale, 8 of them are employed in big scale companies while 7 of them work in medium level and 5 of them work in small scale companies.

The interviews were held between October of 2014 and May of 2015. This process took approximately six months including transcription. Even though I had not planned the field research to be this long; reaching the potential respondents and arranging interviews with them lasted longer than I had expected. Everyone I contacted with wanted to be helpful, however, due to their intense working load, organizing the meetings was a difficult process. As a result, the whole process of field research took longer than planned.

I conducted interviews individually. Despite variations, first interviews lasted approximately 2 hours; although some lasted longer depending on the respondents. However, interviews I conducted later were shorter, that is, they took around 1.5 hours.

8 of the interviews were conducted either in the workplace of the developers or in the cafes located in the same buildings. In 4 of these interviews, I had to chance to visit and observe their offices. Offices were comprised of cubicles. Software developers were usually working on their computers with headphones on. Therefore, it is hard to mention about distinguishing characteristics of software regarding to workplace. They were no different from any other company performing computer-based work.

Only one office was distinct in this sense. In this office, walls were painted in different colors and it was furnished in a more stylish way. During the time I was there, I witnessed a couple of drones flying in the office. 2 of the employees were playing game on a big game console. I can tell that this office represents some features of a newly burgeoning culture of start-ups in Turkey.

Most of the interviews were held after working hours. Except those who work for corporate companies, software developers did not have any strictly defined dress code. They were mostly casual in style. They are mostly dressed in a way to feel comfortable.

1.4 Outline of Chapters

This thesis consists of five chapters. In this chapter, I have outlined the main objective of the study as well as the significance of the object of study. Methodology used to conduct this research has been presented with its scope and

limitations. Moreover, the main theoretical framework has been specified with particular reference to flexible accumulation regime and labor process theory.

In the second chapter, Technical Organization of Software Development Process, I present a general overview of how software development production has transformed in the course of time as well as how technical organization shapes the production relations. Exploring means of production is critical in the sense that through this very examination the production relations are addressed. To grasp these relations, I interrogate influential means of production, namely, programming languages, programming paradigm, programming tools/technologies and use of internet on the basis of its impacts upon production relations in labor process of software development. Parallel to this, I have also discussed through which ways means of production are transformed so as to increase the productivity of workers; hence, the rate of profitability of capitalists.

In chapter 3, I dwell upon the relations of control between management and software developers in the workplace. Strategies of control employed by managers in order to increase and secure surplus value production are delineated as well as the workers' response to those. I also address peculiar characteristics of control relations in this specific field of production. All of these issues are be discussed with reference to the experiences of the workers through their own narratives.

In chapter 4, the role of skills and its effects to labor process of software production are analyzed. The target of this analysis is to reveal how software developers experience job insecurity, and how they develop strategies to endure concern of uncertain future. In this respect, I delve into reasons of uncertain future that exacerbate job insecurity fear among software developers. This chapter also goes into details of from what perspectives skills hold importance for employers and employees along with findings from field study.

Finally, chapter 5 will propose a summary of the evaluation of the findings of the research.

CHAPTER II

TECHNICAL ORGANIZATION OF SOFTWARE DEVELOPMENT AND MEANS OF PRODUCTION

In modern industry, instruments of labour and technical organization of production not only reveal the degree of development to which human labour has attained, but "they are also indicators of the social conditions under which that labour is carried on" (Marx, 1992, 285). That is to say, technical organization of production is one of those points at which capitalist relations of production is produced and reproduced.

The conditions of software labourers have been varied remarkably in accordance with the technical organization of production and its instruments that have been used. Motivated by this observation, in this section, I aim to understand the effects that means of production have on the works of software developers in specific settings of today, by making use of the indications gathered through a fieldwork.

It is widely argued that there has been a significant shift in the organization of software development production. According to Kraft and Dubnoff (1986, 194), it has been shifted from the status and autonomy of the skilled artisan to the regimented task-structures of Taylorised clerical labour in recent years. In other words, software production has been reorganized and simplified either directly by management or indirectly through transformation of the means of production (Beirne et al. 1998). In either way, it has been moved away from craftsmanship-like production process to a more conventional industrial one.

In his detailed work, Kraft (1977) analyzes the instruments of labor to comprehend through which ways idiosyncratic software development process was transformed into a systemic and standard one. By this, Kraft (1977, 20) means, requirement of average skill for developers tended to decrease whereas control of production has increasingly been taken over by the managerial positions. For Kraft (1977, 22), development of software production tools and methods are the principal elements that enabled such transformation of production organization.

Technical organization of production in software development process is key to understand dynamics of relations of production. Rather than merely technical components of production, I think, tools of production and corresponding production organization are indicators of social relations under which labour is carried on. They have great impact upon enabling technical organization of software development in accordance with interests of capital.

However, elements of technical organization of production considerably diversify depending on the variables of product. Thus, instead all of them, I am going to focus on specific elements whose influences upon production organization are deemed to be significant.

In the following parts, I am going to discuss the effects of specific technical organization of production in the sense of cheapening labour power as a commodity and bolstering management control over labor process of software production. Some instruments of production which are presumably more effective in technical and social organization of production and their historical development will be discussed. These instruments of software development are as follows; programming languages, programming paradigms, production technologies/tools and internet in the context of knowledge sharing practices.

2.1 Programming Languages

One of the essential instruments of software development is the programming language which enables the communication between the computer and the software developer. Computers are programmed by these languages so that they can respond to diverse requests of its users. Hence, these languages, designed and used for communicating to the computers, are called programming languages.

In the early phases of modern computers, programming was carried out through the use of a programming language which is based on binary system. This language, which is called as machine language or machine code, is solely based on numeric representation. Therefore, software specialists were using the machine language which could be read directly by computers. Programming on account of using the machine language was rather difficult since it required plenty of knowledge and technical skills.

The second generation programming languages, developed after machine language, contained specifications with a relative level of abstraction. Along with the numeric representation, these languages also made use of some alphabetic symbols for programmers. Still, the second generation programming languages used to have specifications similar to machine language. Thus, languages of this generation together with machine language were defined as low-level programming languages because of their proximity to hardware.

The third and fourth generation languages, which are defined as high-level programming languages, had a profound impact on the course of software history. These high-level programming languages could be functioned independently from the architectures of computers, therefore software programs have become portable

components of computers. More importantly, through enabling the use of words and syntax structures from natural languages, these languages have relatively simplified programming. That is to say, programmers were able to learn and use these programming languages more easily in comparison to more complex, and hard-to-learn low-level programming languages. For these reasons, high level programming languages were defined as more machine-independent and more programmer friendly. Today's widely used C, C+, Java, and Javascript languages are from these generations as well.

Evolution which programming languages have been through can be taken into consideration in regard to its effects on technical organization of software development. Although we will not be able to discuss all details and steps of this development, effects of transition from utilization of low-level programming languages to high-level programming languages are going to be evaluated in more general terms.

Programming languages are used to write programs that communicate instructions to machine, particularly to a computer. In early years of computing, these instructions used to be written in "machine language", based on binary code that is extraordinarily complex and difficult to prepare. Consequently, developers had to have high qualifications like mathematical expertise to meet demands of production.

In the course of time, symbolic "source" languages have been developed through which directions were given in alphabetic and decimal code. Via special software programs (compilers and interpreters later on), the computers started to be used to translate this source code into machine-language. Therefore, it became possible to do programming with programming languages which are more akin to natural language without mastering more complex and difficult machine-language.

This complexity and difficulty of programming languages was an obstacle to the capitalist control of labor process. Because knowledge, thereby programming practices which are hard to obtain was giving programmers a greater autonomy and control over production processes.

Each generation of programming languages at all levels brought about new features to reduce complexity and difficulties of software development. For example, third generation programming languages involved automation of programming. Enabling use of words and syntax structures from natural language's, these generation of languages like COBOL, FORTRAN were intended to replace mathematical manipulation with English-like statements (Kraft, 1979, 176). As a result, programmers are saved from direct contact with the machine code, which started to be generated through software applications called as compilers and later interpreters. Once the task of programmers have been automatized by these applications, required skills and knowledge from software specialists have changed. In the first place, they did not need to know machine code which is hard to learn and implement in order to do programming anymore.

> Language evolution removed the need for programmers to have mathematical knowledge or even knowledge of computers. Although the third generation languages (3GLs) of the late 1950s, such as FORTRAN (Formula Translator) or COBOL (Common Business Oriented Language) were fairly esoteric with precise vocabularies, commands and syntax and therefore still prone to human error, the fourth generation languages (4GLs) of the 1980s, had a vocabulary and syntax more akin to natural language. The evolution of these high level languages made programming more accessible to a wide spectrum of people. (Barrett, 2001, 24)

As indicated above, the more advanced the programming languages have become, the less necessities of production have been put forward. Therefore, average skills and knowledge prerequisites for individual programmers have changed and decreased. Barrett (2001, 31) reach the same conclusion: utilization of advanced

languages led to increasing productivity of labor through automation and simplification of programming. Therefore, evolution of programming languages has tended to reduce complexity and abstractness of programming. Consequently, two managerial goals have been actualized; average skills required for production from labor have been decreased and simplification of programming has led to a relative increase in productivity of labor (Kraft, 1977, 58).

However, given the other instruments of production today, programming languages alone seem to have lost significance over technical organization of software development. That is to say, effects of programming languages over technical organization of production cannot be evaluated without other means of production like programming paradigms and production technologies. One of my interviewees, Cemil declares similar view when we are talking about determining power of programming languages on a software project:

> Çok önemli değil. Dilde uzmanlaşmak çok bir şey ifade etmiyor. Türkiye'de herkes ya ben şu dili biliyorum, bunu biliyorum diyor da proje yapmak öyle bir şey değil. Genelde bir proje yapmada, dilin etkisi %10'dur. Kalanı perspektif, mimari, tasarım, thresholdlar, optimum noktaları bulma, mühendisliğin kendisidir yani. Esas zaman alan budur. [Cemil, 34][2]

I do not think Cemil was trying to say that programming languages are not important for the operation of software anymore. Rather, he emphasizes the fact that focus seems to be slided towards design aspects of software production as scale and complexity of software applications are increased. According to him, efficiency and

[2] It's not that important. Having expertise on language alone doesn't mean anything. Everyone in Turkey says "I know this and that language" but conducting a project has nothing to do with it. Generally, influence of language constitutes only 10% of a project. The rest is having a perspective, architecture, design, thresholds, figuring the optimal points, I mean, it is the engineering. This is what really matters, what you spend your time on. [Cemil, 34]

operability of software application is dependent upon architectural structure of application today more than its constituents.

That is actually true, for most of my interviewees as well, architecture is more determinative over software applications' operations. However, it does not necessarily mean that expertise on programming languages is trivial. I think Cemil's words imply another thing about programming languages.

With programming languages which include syntax structures and words from natural languages, doing programming has become much easier. It is clear that with these languages, programming is available for a wider spectrum of people today. We know that even kids in primary schools can learn these languages and write simple programs. So, Cemil's quote might also be taken as an evidence of how utilization of programming languages have become transparent within software development. Professional software producers, therefore, tend to see expertise on programming language as taken for granted. And they are prone to highlight less known, thus distinguishing aspects of programming like programming paradigms, prominent features of software development life cycle.

Even programming languages are used today seem to have lost their influence over production organization, the evolution of languages over time is a valuable guide for us to see the dynamics that shape technical organization of production. As I will discuss, other instruments seem to be developed and implemented in production point in a similar fashion: simplifying the production process and decreasing the dependency of employers on workers. In the following section, I am going to analyze another instrument of production: programming paradigm which had a great impact upon development and concentration of conceptual aspects of programming in terms of management.

2.2 Programming Paradigms and Standardization of Production

As the variety and the complexity of the tasks handled by software applications have increased, the questions of how an application executes these tasks, hence how it is designed have gained more significance. The design principles have been developed concerning the technical requirements and demands which software application has to respond. These body of rules/principles, which determine how the program operates its tasks are named as programming paradigm. Specific paradigms are produced to be met with specific needs of software application. Procedural programming, structured programming, object-oriented programming, and declarative programming are some of these specific paradigms.

Until the programming paradigms became widely used, programmers would use their creativity, knowledge, and skills in order to resolve the problems they encountered in programming. They would come up with their own resolutions, and concepts. However, as the use of programming paradigms became prevalent, the field for developers to perform their mental activities had relatively shrinked.

Programming paradigms have enabled the standardization and modularization of software development process. As the production becomes more fragmented, programmers have become able to work independently from each other. From this perspective, paradigms have been influential not only about how a software application should execute its tasks, but also how and in what ways software developers should carry out programming.

I must state that, our intention in this chapter is not to discuss instruments of production alone in their technicality but their specific settings of utilization within capitalist relations of production. While evaluating machinery in modern factory system, Marx (1992, 544) considered necessary the distinction between settings

where the collective labourer is the dominant subject and the machine is the object, and in which the machine is the subject and the workmen are merely conscious organs subordinated to central moving-power of it. According to him, former settings of labour division over time has been thrown overboard by the latter in the factory. That is to say, in capitalist production, the workman does not employ the instruments of labour, but the other way around. I think programming paradigms can be taken into consideration in this sense, regarding their increasing influence over technical organization of software production.

Programming paradigm is generally defined as to bring design principles, rules for the internal operation of software application. Based on that, one can say that it is just another instrument of production which software developers make use of in their activities. However, situation seems to be the other way around regarding current software production organization. Because, software development practices today reflect overwhelming influence of the principles of programming paradigms more than the autonomy and discretion of labourers. So, it would not be wrong if we propose that the software producers, to a certain extent, are turned into the extensions of programming paradigms. They are the ones to show adaptation to the imperatives brought by programming paradigms. Based upon that, in this part of study, I will look in what ways introduction of programming paradigms have had effects on the work of software developers' labor.

Programming paradigm, a fundamental computation style of computer programming, can be traced back to the very early days of computing. It is used to serve as a way of building structure and elements of computer programs. Specific forms of the computation to be performed by program are defined through them (Philipson, 2005, 16). To put it differently, programming paradigms determine the ways how a program carries out its tasks.

Today, there is a great variety of programming paradigms. Thus, it is difficult for this study to look each one of them to evaluate their effects on production process. Instead, I have picked the ones whose effects are more conventional and fundamental for contemporary software development. My goal is to see whether or not there is a connection between the discussions within the Labor Process Theory and the functioning of programming paradigms.

In the beginning, there were few conventional paradigms that defined the operation of programs. Therefore, programmers had more discretion and autonomy over their work. This autonomy of programmers, hence their idiosyncratic way of programming rendered them irreplaceable for the production process. This situation was posing a huge threat to the continuation of production since all project/program would have been ruined if programmer in charge quit the job. In these cases;

> Managers and owners are vulnerable to all sorts of calamities. Programmers can fall off mountains and die. They can quit and go to work for a competitor. Or, they could simply demand more money or changes in working conditions-and get them because of their indispensability. (Kraft, 1977, 57)

This was an obstacle for companies to reduce or eliminate uncertainty in the expenditure of labor while the same time guaranteeing the production of profit. That is to say, managers had to reduce the cost of uncertainty caused by the worker while at the same time maintaining the profit. For this reason, in the course of time, utilization of paradigms have become conventionalized to diminish dependency of management and capitalist on individual programmers. Rationale behind this was simple: in order to maintain control over technical structure of program thereby over the production process, there had to be some standards, formal procedures through which production can be systematically controlled and monitored. Programming

paradigms met this managerial demand and promulgated some standards to be followed by programmers.

As utilization of paradigms, with assistance of other instruments of production, has increased, dependency of capitalist on software developers decreased to some extent. After programming paradigms have considerably regulated the main structure of programs, hence of the programming, it has become relatively easier to substitute worker with another. One of the aims behind standardization of production was securing the contiunity of production. As it is evident in Emrah's quote:

> Kodun daha derli toplu olması gerekiyor. Önceden herkes istediği yere istediği koda, orta yere dan diye dalıp burada bu çalışacak diyordu. Artık öyle değil. Hem bu teknolojiler seni daha temiz kod yazmaya zorluyor, öyle de olmalı. Çünkü bir adam bir şirkette senelerce çalışmıyor. Yarın gidecek, başka birisi gelecek, gidenin kodunu gördüğü zaman orada hangi mevzunun döndüğünü anlaması lazım. [Emrah, 25][3]

According to Emrah, production paradigms and standarts in any other form should be followed by workers, in order to eliminate any possibility of production interruption. In the factory system part of Capital, Marx (1992, 546) depicts similar situation related with machinery and its specific settings in capitalism. According to him, as long as the motion of the whole system does not proceed from the workman, but from the machinery, a replacement of workers can take place at any time without an interruption of the work. It can be told that production paradigms in a similar way determine the process of software development and the limits of what programmers can do and cannot.

[3] The code needs to be neatly written. Previously, everyone could intervene in the code as they wanted to. Now, we can't do this. These new technologies compel you to do coding better structured. This is how it should be. Because no one works in a firm for long times. She will quit job soon and a new employee will start. The new one should be able to understand what her previous colleague writes. [Emrah, 25]

In his detailed study, Kraft (1977, 56) asserts that emergence of structured programming paradigm in 1960's led to radical break within the history of software development. This paradigm brought about the introduction of some elements like control structures (sequence, selection, iteration), subroutines and blocks into structure of programs to achieve simplicity, quality and development. Orderliness through more efficient control structures as well as economical use of standard languages and code were stressed to avert programs compounded by hardly-readable, idiosyncratic "spaghetti" code.

As a result, structured programming paradigm made it possible that the worker as a producer can be ignored to some extent; that management becomes the designer and that product is comprised of its plans, procedures and instructions mediated by instruments of production. That is to say, generalized knowledge of production which individual producers used to have, came to be gradually extracted and gathered by the managerial positions. They were incorporated into the process with the help of paradigms, on the condition that it meets demand of management.

However, effects of structured programming paradigm were not limited to ensuring the reproduction of labor and production. Standardization of product also rendered disassociation of production into its elements an applicable option. If a product - software program in our case- can be standardized, it can also be produced in a standardized way by people who do the same limited tasks over and over without knowing how they fit into larger process. As articulated by Kraft;

> Structured programming and modularization thus achieved two long cherished managerial goals at once. They freed managers from dependence on individual high-level software workers. They also made possible for the first time a genuine job-based fragmentation of labor in programming. (Kraft, 1979, 148)

Dividing program into discrete elements has made it practicable to assign these software elements as well-defined tasks to individual workers. Translated into market terms, as the software element is divided into smaller tasks, the labor power capable of performing this task gets cheaper compared to labor power of a more skilled, integrated capacity worker (Braverman, 1998, 57). For this reason, it is claimed that structured programming and its logical partner, modularization is "the software manager's answer to the assembly line, minus the conveyor belt but with all the other essential features of a mass production workplace" (Kraft, 1979, 59).

The development of object-oriented design [OOD] and programming in 1990s can be seen as an extension of structured programming paradigm in terms of its effects on the software development. As a successor of it, the OOD has increased modularization in software projects. It enabled grouping of the different elements of program into classes with different areas of functionality or data objects within software application. Thus, identification of discrete objects in a layered structure within the system has become easier with the design principles of the OOD (Quintas, 1994). Gökhan's description of OOD is also explanatory in this sense;

> 90'lardan sonraki en büyük değişiklik paradigma değişikliği, object oriented analysis design sayesinde oldu. Devasa projeler yapılmaya başlandı. Building block dediğimiz yapılar daha modüler hale geldi. Modülleştiremediğin dönemde tanımladığın değişkenler düzensiz ve saçılı bir halde dururdu. Sisteme bakan bir yazılımcı da ondan etkileniyordu ister istemez. 1000 tane değişkeni, onları nasıl kullanacağını, her şeyi masaya saçılı haldeyken düşün. Algı karmaşası, insan algılayamaz bunu. Ama masanın üstünde 20 tane küçük kutu olsa, üstünde de bir etiket olsa sadece, basit bir taramayla bulabilirsin ihtiyaç duyduğunu. şayet isimlendirmeyi de iyi yaparsan, aynı anda ilgilenmen gereken şeyin sayısı ciddi ölçüde azalır. [Gökhan, 44][4]

[4] The most significant paradigm shift since the 1990s has been the "object oriented analysis design". Huge projects has been started to be conducted with it. Building blocks have become more modularized. When you were not able to modularize, all elements of software

Gökhan's comment on how the OOD transformed the software development seems to support our claims about the effects of programming paradigms. As quoted below, implementation of programming paradigms, particularly the OOD programming, has led to concentration of conception part of production in the hands of management, or of "software architecture", fragmentation of project into logical units that developers can work on independently from other units and workers on the other.

> Mimarın başarısı, yazılım mimarlığı diyoruz o göreve, o role, onun başarısı da, projenin büyüklüğüne göre bu enkapsülasyonu en pedagojik şekilde yapmaktır. Bugün encapsulation, information hiding bu yüzden çok çok önemli bir seviyeye geldi. İnsanlar da birbirinden bağımsız logical unitler halinde mantıksal üniteler içinde çalışıp, diğer insanların yaptıklarından en az etkilenecekleri bir yapı arayışındalardı. Object oriented tam da bunu kapattı. [Gökhan, 44][5]

In conclusion, advancement of programming paradigms, hence modularization and standardization of software programming renders supervision of software workers easier. It divides a program into its discrete units, then makes it possible to allocate them to the workers specialized on that particular division of production. In the end,

---

application were stored in a messy and unorganized way. A developer which works on such system had been affected by this. Think about you look at your table and see 1000 different variables which every one of them is crucial for software application. You can't understand what is going on. But if you see 20 different boxes with name stickers on them, you can find whatever variable you are looking for. If you classify them well, with proper names, then you can easily find everything. The number of things you have to deal with decreases drastically. [Gökhan, 44]

[5] The success of the architecture –we call software architecture- is to do the encapsulation in the most pedagogically convenient way. Think about 1000 objects with 10 different elements in each. Within this code, there will be 10.000 features. It used to be impossible for a developer to see, maintain, regulate, fix and add some functions. That's why, encapsulation and information hiding have become extremely important. People were also looking for a structure which could operate as independent logical units and which would not be affected by what other developers have done. Object oriented programming achieved this. [Gökhan, 44]

it becomes possible to regulate the production and the workers through this programming paradigm itself, in a more mediated and automatic way.

So far, we have seen that software development process has been simplified and fragmented with the advancement of instruments of labor. Generalized knowledge of production has been appropriated from the productive workers and concentrated in the hands of managerial positions. Software development process has been restructured and redesigned in such a way that knowledge of production is kept in compliance with the managerial interests while decreasing the significance of the workers' productivity, Hence, dependency of the employer on the workers decreases.

From the perspective of capital, refinement and reproduction of knowledge of production is critical to the production process. However, traditional ways of its transmission from a person to another is costly since it requires considerable amount of time. Instead, extracting it from workers and putting it in compatible forms can diminish the costs of reproduction and exploitation of it to a greater extent (Perolle, 1984, 114). As the knowledge and skills extracted from productive workers are accumulated within the instruments of production and organized along with capitalists' interests, reproduction of knowledge can be more efficiently achieved. Therefore, skills and knowledge demanded from individual worker as well as dependency on the workers decrease. Tools and technologies of software development, in this regard, is another element of software production which should be analysed.

2.3 Effects of Production Tools/Technologies on Simplication and Automation of Software Development

Today, software programs are written on software platforms by using complex software packages; guidance is provided through these programs. One of these, Integrated Support and Development Environments consists of source code editor, automation and debugging tools. In addition, application-specific software frameworks serving generic functionality to users are very critical and indispensable parts of production.

In this regard, software development and support tools, technologies -firstly defined as Computer Aided Software Engineering [CASE] tools in the late 1980s- are used to insert knowledge of production as an input. Great amount of knowledge or practices that would have needed years of working experience for developers to get is now easily available with the help of these instruments (Berry, 2011, 36).

Introduction of these tools has been used to simplify programming and increase control of management over development process. In this sense, evolution of software development towards simplification and routinization of programming are evident in testimonies of my respondents who have known both the older and current technical organization of software production, therefore who are aware of the differences between two.

> 1970'lerden 1990'lara kadar workstationlar, sun workstationlar, üniversitelerdeki araştırma bilgisayarları diye tabir edilen şeylerde insanlar binlerce satır kodu, tek satır tek satır daktiloda yazar gibi yazıyordu. Ne değişti, şu an ne farklı ? Bu aslında şöyle bir şey oldu; Bir satranç oyunu gibidir program yazmak, eskiden öyleydi. Resmen düzinelerce satır kod yazardık, onu çalışacak diye makinaya sokardık. Makina bize şuralarında problem var diye dönerdi, anında söyleme yoktu. Paradigmaların bu kadar basitleşmesi, IDE platformalarının (Integrated Development Environment

> Platforms) bu kadar akıllan-masıyla programlama da ciddi şekilde değişti. Yeni IDE'ler artık yazdığın anda guide ediyor, nereye gideceğini söylüyor. [Gökhan, 44][6]

This simplification and fragmentation have indicated its effects firstly at the expansion of pool of available labor power. Gökhan and Yüksel share the view that doing programming has become available for a wider spectrum of people since it is being simplified with the advancement of instruments of production.

> Dili biliyorsun, herşey burada, kitaptan da bakabilirsin ama kodu yazarken notepad' de yazardın eskiden, kütüphane kafana tamamen oturmuş durumdaydı. Şimdi o düzeyde beyin faaliyeti gerçekleştirmeye gerek yok. Şimdi öyle bir şey yok, yanındaki guide, yazılım geliştirme araçları yanlış yazdığın anda altını çiziyor, nasıl Word'de yanlış yazdığın bir kelimenin altı çiziliyor, aynı şekilde düzeltiyor seni. İşte bunlar hep kullanılan araçların çok gelişmesiyle gerçekleşti. [Gökhan, 44][7]

> Ben hep söylüyorum bunu, yazılımcılık, hackerlık eskidendi. o zamanlar insanların elinde sınırlı kaynaklar vardı. bunlarla bir şeyler yapmaya çalışıyorlardı, daha fazla kafa patlatıyorlardı. Şu an ise kullanılabilecek kaynaklar sınırsız. çok fazla programlama dili seçeneği var. içinden seçip uygulatabiliyorsunuz. Eskiden böyle değildi, hakkaten kafa yoruluyordu, bunu daha hızlı, efektif nasıl yapabilirim diye. State of art'tı o zaman için

[6] In 1970s, until the 1990s, people had used to do coding as if they were writing with typewriters. They had have to do programming line by line on the work stations or research computers of the universities. So, what has changed, what is different now?. Till a closer past, it was like playing chess. We would write hundreds of lines of code, and then run it on the machine. Machine would tell us about the bugs after a while. But today, simplification of paradigms and improvements on software development environments have seriously transformed programming. Thanks to the new integrated development environments (IDE's), computer can immediately guide us, it gives us directions today. [Gökhan, 44]

[7] You knew the language; everything was written in your mind. You could open the book but while you were writing on notepad, you had to memorize the library. Now, it is easier. Being a software developer is easier now. Previously, it was a sort of brain functioning at a really high level. Only those who were trained, maybe professor, could do it. Now, it is different. Integrated Development Environment (IDE) platforms give you direction; it is the same as Word programs corrects spelling mistakes. This has happened thanks to the extensive development of our instruments. [Gökhan, 44]

> programlama yapmak. şimdi biraz ne biliyim, artık c'de 100 satır yazdığınız şey için library olunca, kimse o 100 satırın nasıl yazılacağını düşünmüyor. [Yüksel, 40][8]

As indicated in the quotes, some parts of production are simplified and automatized; therefore, contrary to the past, developers are enabled to execute tasks assigned to them without the obligation of having detailed knowledge of production. Intellectual activity of software development in the sense of pondering conceptual aspects of programming does not require that much time and effort any more.

Description that Yüksel and Gökhan have made above also implies the internal polarization between conceptual and executive parts of software development. As programming got easier, division between critical conceptual tasks and routine executive tasks has been sharpened. During stages of software development today;

> Only a small minority formulates problems, makes the decisions, and takes appropriate action based on information provided by the computer. Most of the routine and tedium associated with the daily tasks of producing programs are left to an anonymous army of people who merely do what they are told to do, understanding little of what they do and less of why they are doing it. (Kraft, 1977, 29)

In this sense, new generation of developers' lack of detailed knowledge of software development, implied by Yüksel in above is a natural outcome of transformation in technical organization of production rather than laziness or lack of enthusiasm of them. What actually happened is that software development process which was once

[8] I always tell about this. Being a software developer or a hacker, these are bygones. Back in time, people used to have scarce resources. They were trying to achieve somethings, they were dwelling on these issues. Now, new generation of developers have unlimited resources. There are huge variety of programming language. You can pick up one and run it. Previously, you would have to think about how to do it effectively. It used to be a state of art. Now, I don't know, it is easier. You can find 100 line of coding on C in the library. Therefore, no one thinks how to write this 100-line of coding. With one single instruction, they find the code which has already been written. [Yüksel, 40]

acknowledged as craftsmanship has lost some of its artisanship character, at least for the developers composed of mostly new generation of young people.

Generalized knowledge of production has been extracted from individual workers and distributed to workers in the form of partitioned tasks. Therefore, craftsmanship that requires intricate knowledge of the job, the use of tools, materials as well as the process of the act which has been gained throughout the work has been lost. Necessities that production required have started to be provided by instruments of production which is partly owned and controlled by capitalists rather than individual workers. As a result, activity of workers at the site of production has appeared to be animated by the management or instruments under its control (Braverman, 1998, 94). While comparing older and current practices of software development, Yüksel emphasizes this tendency towards degradation of software development practices.

> Eskiden insanlar kafa yoruyorlardı, bir şeyi yaptıklarında neyi neden nasıl yaptığını biliyordu. Şimdi pek çoğu neden yaptığını bilmiyor. Orada bir fonksiyon var, ben onu çağırdım, o benim için yapıyor diyor. Yapamadığında neden yapamadığını bilmiyor ama. Tecrübe yine devreye giriyor mesela. Ben o fonksiyonun yapamayacağını, neden yapama- yacağını biliyorum. Ama benden sonraki kuşak bilmiyor. Ben şimdi bir yerde emekli olacam. Benden sonra gelen grup da bir şeyler yapacak. Ama sonraları gelen nesil, tamamen bilinçsiz bir şekilde çalışıyor mu çalışıyor, çalışmıyor mu ha o zaman bilmiyorum demeye başlayacak. İnsanlarda tembelliğe doğru, yazılımda da fabrikasyona doğru bir trend var. [Yüksel, 40][9]

It should be noted, however, totality of software development still requires complex and difficult process. That is to say, as the intellectual work was divided into design

[9] Software developers used to dwell on what they were doing, why they were doing. Today, many of them have no clue of why they do. They think, "there is a function which I have called, it does everything for me". Once the function fails, they don't know why it fails. At this point, experience kicks in. I know what this function is capable of and what it is not. But the next generation doesn't know. I will be retired at some point. But, new generation will do this job in complete ignorance if the function works or not. People lean towards laziness; in software, fabrication is the general tendency. [Yüksel, 40]

and execution tasks, this process simultaneously produces both skill enhancement and deskilling. Therefore, increase in complexity of general production on the one hand and simplification of discrete units of project allocated to workers on the other, does not necessarily contradict with each other. On the contrary, these two are mutually being reinforced by producing each other.

According to Braverman (1998, 294), what is deskilled is not the entirety of production, rather it is the skills and knowledge demanded from individual workers. In other words, the introduction of tools and procedures organizing and controlling process renders big scale projects possible, while the same tools and procedures enable fragmentation and simplification of production. In this regard, development of technical organization of production has intensified, not diminished, the separation between those who think and those who do everything else, a division which is now beginning to separate software workers as well (Kraft, 1977, 29). Gökhan's quote below reflects correspondence to such polarization of workers between more critical, conceptual tasks of production and less critical, more executive tasks.

> Sen IDE'yi çok geliştirsen de yine critical missionlar var, average olanlar var, ortalama kritik olanlar var. İşin kritikliğine göre kalifiye elemanı bulman gerekiyor. Ama veritabanı yapar mısın kardeşim, tamam yaparım diyor, bir şey de yapılıyor hakkaten. Ama sonra başına kalabiliyor. Adam seçmek zorundasın, her işte böyle, orada sistemi değiştirecek halim yok bundan hoşlanmasam da. Kişi kalitesi azalıyormuş gibi, hep görünür ama bu, öğretmenler eskiden böyleydi öğrenciler böyleydi, bizim zamanımızda şöyleydi gibi bir sürü şey söylenir. Ağlanır, mızmızlanır, ama bu böyledir, çünkü işler giderek kolaylaşır. [Gökhan, 44][10]

[10] Even if you develop IDE as much as you can, there will be some critical missions, there will be mediocre ones as well as ordinary ones. You have to find the worker in accordance with the level of significance. But it is different here. You ask your job applicant: "would you do database?", she accepts and eventually completes the work. But after a while, this job fails. You have to pick up the employee very carefully. This is the same in other fields. Even if I don't like this, I can't change the whole system. It seems as though there are less

In the light of this, it can be proposed that the hierarchical gap in the technical division of labor, as well as the confusion of those in the middle increases as a result of the advancement of production instruments. Incorporation of these sophisticated instruments deepens Braverman's account of the separation between conception and execution parts of production.

This is evident in the polarization between workers; those whose labor power is essential on the one hand and those whose labor is interchangeable on the other. Ceren's views clearly express how the utilization of instruments of production which simplifies and automatizes production has an impact on developers' working career. Her response implies that polarization has got sharpened, influenced by the production instruments, depending on the developers' relation with them. For her, skills and knowledge one can have tend to decline, so does his/her chance to be employed for more critical positions within the production as reliance of a developer on these production tools increases. Therefore, the influence of instruments on production outweighs effectiveness of individual developers.

> Bir de genelde benim işim kökten yazılımcılık diyim artık, onlar abstract çalışıyorlar. Öyle bir fonksiyon varmış, onu çağırıyorlar, onu kullanıyorlar, parametre giriyorlar, bu. İçini, algoritmasını, perfor- mansını bilmiyorlar. Artık yazılım işi biraz abstract olmuş. Fiziği bilmek zorunda değiller artık. Sadece çağırıyorsun, mesela destroy object diyor, class'ını yazıp, .destroy diyorsun, destroy ediyor, bu kadar. Adam nasıl destroy ettiğini bilmiyor. Geliştirdiğimiz yazılım uygulamasının performansı uygulamanın kullanılacağı yer için oldukça kritik bir öneme sahip, bu yüzden benim bütün bunları bilmem gerekiyor. Bir yandan da iyi bir durum bu, ben ileride Nvidia'da çalışabilirim ama onlar çalışamaz. [Ceren, 25][11]

---

and less qualified employee. This was the same in the past. Everyone complains about this, because the work itself gets easier. [Gökhan, 44]

[11] My work is "pure software development" I guess. The others usually do abstract works. They use pre-made functions, enter parameters. They don't know how it works, what is the

It was very clear in our interviews that younger developers share similar views and feelings about their working practices. However, as "exempted from" the knowledge of older practices of software development, they emphasize more on tedium and exhaustion caused by their work content. What they strongly put forward in their statements is the routine-like and easy-going character of their work:

> Bir yandan kafa işi diyoruz ama bir yandan rutini de var. Hep aynı şeyleri görüyorsun. İşlerin %80'i de zaten, 'create, read, update, delete', duymuş-muydun hiç? Kullanıcı verilerini alıyorsun database'e koyuyorsun, sorunca göste-riyorsun, gruplayıp gösteriyorsun. Yazılım dediğin şeyin büyük bir kısmı bu zaten. Acaip algoritmalar, ilginç ilginç şeyler değil. Bir memuriyet gibi bişi bir tarafıyla. Bir takım yazılarla uğraşıp, evrakla uğraşır gibi çalışıyorsun. [Hakan, 26] [12]

> Türkiyedeki bilgisayar mühendisleri için söyleyebilirim, kimse aman aman bir şey yapmıyor. Bir kod var, bir kod yazıyorsun, bir java yazıyorsun. Database'e girip sql ile bir şey aratıyorsun. Bu yani, bir şey yapmıyoruz. İnanılmaz şeyler değil. Şu an benim yanıma birisini verseler, adam değil bilgisayar mühendisi yeni mezun hiç bir şey olmasın, lise mezunu ortaokul mezunu olsun, ben bildiklerimi o adama 45 günde yaptırırım. Burada mevcut çalışan bir şeyimiz var zaten. Ona eklemeler çıkartmalar yapıyoruz.

---

algorithm, how is the performance. Today, software development has become more abstract. You just call the function. No one has to know about the physics operating behind it. For example, she gives the command "destroy object", she writes the command and it destroys itself. That's it. She doesn't know how this function destroys. In my job, I need to know all of these things, because performance of software application is very critical. On the one hand, this is very valuable for me. In the future I can work for Nvidia, but the others can't. [Ceren, 25]

[12] We assume this is a kind of mental labor but it has its own routine, for sure. You always encounter with the same things. 80% of the works we do is "create, read, update and delete". Have you ever heard of it? You receive the user data, insert them into the database. If anyone asks, you will group them and show. This is biggest part of software development. Rarely you encounter with strange algorithms, eccentric things which you have to deal. It is kind of civil service work. It is like you are working by tackling with paperwork, documents. [Hakan, 26]

> Ben bu adama 15-20 gün kodların ne işe yaradığını anlatsam, 1 hafta da şunları düzelt desem, bu adam 1,5-2 ay içerisinde bir şeyleri yapabilecek duruma gelir. [Fırat, 30][13]

> Hep aynı, kopyala yapıştır, kopyala yapıştır. Yeni bir şey geliyor, onun kodunu yazacaksın, o nasıl tanımlanır, bir özellik tanımlaman gerekir. Öncesinde yapılmış işlerden o özellik nasıl verilmiş, şöyle verilmiş. O isimleri değiştirerek yeni dosyaları oluştur, onun içine kopyala, class isimlerini değiştir, yeter. Bunun için programlama dahi bilmen gerekmiyor, okuma yazma bildiğin zaman bunu yapabilirsin gibi bir durum var. [Leyla, 25][14]

Transformation of production organization towards simplification and standardization has clear impact over social relations of production. As Burawoy (1985) asserts, relations in production and relations of production are reciprocal aspects of production process. Thus, changes in one substantially affect the other since they are mutually operated.

These developments which have caused cheaper labor and increasing control over production process have detrimental impacts on the workers, particularly in a sector within which wages are relatively high; whereas, the output of production carries great risk. Therefore, the bigger the reserve army of labor, the less employment

[13] I can say that most of the computer engineers in Turkey are not doing programming which is really challenging or creative. You always have a ready project, you are working on its codes, you are always using similar languages, writing with Java for example. You are entering database and search something with SQL. That's it, we are almost doing nothing. At least, these are not difficult or incredible things. If they gave someone to my guidance, I could teach everything necessary for what I am doing at work to him or her. I would do it within 45 days, even s/he is graduated from primary or secondary school. [Fırat, 30]

[14] It is the same, copy-paste, copy paste. When the new task is assigned, you need to define it first. If you realize the task is similar to your previous assignments, and it generally is, you define a property for new one based on previous works. You just create new folders, copy codes into folders and change the class names, that is enough. You don't even have to know programming, for doing this, knowing how to read and write would be enough. [Leyla, 25]

security is guaranteed for the workers. Therefore, the conditions that led to employment insecurity and precariousness is closely related to the transformation of production process.

To Magdoff (2004), reserve army of labor concept is a comprehensive instrument to discuss precarious working conditions which is not new within the history of capitalism. For Magdoff (2004), knowledge and skills demanded from individual workers have decreased as a result of simplification and fragmentation of work. Repository labor power, hence the reserve army of labor has expanded; thus, the power of capital vis-à-vis the labor amplifies.

> Regardless of the potential for workers to share some of the income generated by increased productivity, more efficient production creates problems for labor. With greater labor productivity—whether produced by new machines or more effective management techniques for controlling labor ("doing more with less," as they say nowadays)—fewer or less skilled workers are needed for a given level of output. Thus, productivity growth is a constant threat to employment security, especially with respect to well-paying jobs in mature industries in which total output is not growing rapidly. (Magdoff, 2004, 24)

What particularly older interviewees strongly underline during the interviews was the uncertainty and unpredictability of future of production organization. They seem to have confronted with this tension caused by these uncertainties more heavily than their younger colleagues since they can more clearly grasp how production has been transformed and might be in near future. At least, they are aware of the transformation of production organization towards simplification, modularization and fragmentation is a threat to their employment security; maybe not for today but for their future. In this regard, my respondents' resentment about drastic change in production organization and its expanding feature of reserve army of labor can be taken as an evidence to their concern related with insecure working conditions.

> Kızılaya git, 1000 tane bulursun [yazılımcı], o kadar ayağa düştü ki. SO benzeri siteler var, çinli birisi mesela 10 dolara, 20 dolara bilet satış sistemi yapabiliyor. Facebook'un aynısını yaparım diyor, 100 dolara yaptırıyorsun. E yapın madem. O kadar ayağa düşmüş durumdaki akıllara ziyan. Valla bilmiyorum, ben yazılım tarafındaki tecrübem sonucunda, bu işin tam otomasyona gitmesi gerektiğini düşünüyorum, olacaksa o olacak. Bu iş olmadığı sürece yazılımın bir anlamı yok gibi geliyor bana. [Semih, 44][15]

Taking all these statements into consideration, it can be concluded that transformation of technical organization of production strives to cheapen labor power and to transfer control over production process from producers to the management. Its effects on production relations, cheapening labor power, potential expansion of employment pool exacerbate both the tension of insecure employment relations and uncertain future for workers.

Internet, as a facilitator of practical knowledge transmission related to software development is another key element of development process. It is observed in the field study that degree of reliance on internet among software developers is remarkably high. Therefore, effective knowledge sharing practices enabled by internet have reached such a point in the production processes that internet has become one of the major transformers of the technical organization of production as well as production relations.

2.4 Effects of Internet Usage on Software Development

Software development is an error prone process. Developers are expected to deal with these problems which affect the operation of software application immediately as they appear. Despite the fact that they have variety of tools to prevent and correct

[15] You can find 1000 software developer on the street. Everyone can do this job now. There is nothing special left. There are websites similar to StackOverflow. A Chinese person can write a ticket sale system for $10-20. They can imitate the Facebook for $100. I don’t really explain it, this is crazy. [Semih, 44]

these errors, there are always problems which cannot be detected by automation and debugging tools. Hence, programmers have to solve them. Therefore, from time to time, sorting out these problems might become troublesome and time-consuming tasks for them. The programmers with whom I conducted interviews expressed that in such cases, the first resource they resort is the Internet. Today, the information shared on the Internet platforms, and cooperation done through these platforms, have a great contribution to sort out the problems which programmers are faced with. This characteristic of the Internet is the one that makes the Internet one of the most critical elements of software developing in programmers' perspective.

From the first day on, the internet has been a tool used by software developers. Beyond providing the information in a short time, the internet enabled gathering of the groups that can discuss, and share information. Accurately, the use of question and answer websites, at which people help one another to find solutions for almost anything one can produce, has been widespread dramatically. Stack Overflow (SO) is an example of these websites, through which professional or enthusiast software developers collectively look for their software-related problems. Even though it is a website only for the people interested in software, according to August 2015 surveys of Alexa.com[16], SO is the 55th most visited website in the world, and the 45th in Turkey. With regard to the extent of its use, it can be argued that SO, and many other websites alike have a significant role in production process.

Internet simply enables developers to access information whenever and wherever needed. Labor productivity increases, hence, the quality and speed of production is enhanced. For this reason, the utilization of internet as a way of knowledge sharing practices seems crucial to understand production process in this industry. In the light of these, it can be drawn that utilization of internet too, has brought about substantial effects in production organization as advancement of abovementioned

[16] http://www.alexa.com/topsites/countries

production instruments. Thus, in this part, I am going to explore firstly the influence of internet utilization over technical organization of production, secondly, its repercussions on production relations.

Internet is the name of whole infrastructural system providing various kind of services and resources. In this respect, boundaries of its effects on production are wider than I could discuss here. It certainly exceeds limits of my research. Therefore, I confine my research to its enabling features of knowledge production and exchange practices as well as its impacts upon production process. Before discussing my field research, it is important to given an account of how internet has contributed to the paradigm shift in knowledge management systems.

As I have briefly discussed above, programming is knowledge-oriented activity, which requires remarkable amount of knowledge and skills. Therefore, knowledge has always provided a bargaining power in labor and capital relations, especially in information-oriented industries. However, with advent of technologies supporting and facilitating the practices of knowledge exchange, dynamics of bargaining have been reshaped in favor of capital.

Until recently, knowledge acquired by the workers had been the main source of production. Control of production used to pertain to individuals rather than organizations. This was posing a threat to the firms since knowledge needed for production was dependent on the employee. Furthermore, people had more chance to exchange their knowledge for some returns such as raise, reputation, promotion and etc. This was a serious barrier to the distribution of knowledge within companies, hence to the general productivity. Such mechanisms, called knowledge management systems, were developed so as to maintain and spread this knowledge within organizations. Similar to the separation between conception and execution parts of production, the aim was to convert knowledge residing in memory of

workers as much as possible into transferrable structural assets owned by firms, like documents, procedures and etc. (Wasko and Faraj, 2000, 157). Thus, possible loss and conflicts would have been averted.

With proliferation of internet, context of organizational knowledge management has undergone a serious transformation. Online communities have emerged one after another where knowledge transmission can be held like the one among professional or enthusiast programmers. Therefore, third kind of source, knowledge embedded in community has become available for organizations in addition to knowledge codified into structural assets and embedded in individuals. Right after open-source projects outcompeted closed-source alternatives, knowledge management systems within organizations have started to support these mostly cross-firm, community based knowledge exchange practices (Wasko and Faraj, 2000, 161). Hence, platforms, services on internet where knowledge is produced, accumulated and distributed have become substantial resource of software development.

As far as I observed, reliance of developers on the internet, especially in first years of developers' career is excessively high. Due to its contribution to their work, my respondents repeatedly stated their gratitude to the internet. Eralp chose to verbalise this bond as a kind of friendship. According to him for example, 'internet is the best and only friend of developers at work today'. To describe this strong interaction, further scrutiny is needed. Thus, in this section, for a better comprehension of production process, I am going to discuss from which perspectives internet utilization is useful for employers and employees

In spite of the attempts to standardize production as I have discussed in previous sections, software development process is still prone to uncertainty and unexpected problems. Coming across with insidious and hidden problems is a part of developers' working routine. Problems are hidden because, in most cases they spoil

the activity of software without revealing themselves. Putting it another way, the reasons of a failure are hardly visible. Therefore capacity to identify problems, for this reason, holds up its significance in the production process.

Ability to find out problems depends heavily on working experience and it requires certain qualifications. Thus, detection of problems might turn to a painful, weary and time-consuming tasks. My interviewees mostly touch upon difficulties of this kind while stating how internet use is beneficiary and helpful to cope with those.

> Ben her defasında yazarım, çıkan en baştaki 5 6 sayfayı tık tık açarım. Bakarım, uyar, uymazsa kapatırım, bazen alır denerim. Mecbur kullanmak zorundasınız. Bazen öyle çıldırtıcı şeyler oluyor ki, kontrol kontrol kontrol, hiç bir şey yok bunda, neden çalışmıyor bu kod diyorsunuz ama gerçekten araştırıyorsun, bir yerden bir hata çıkıyor. İnternetin en fazla yararlı olduğu nokta da bu, internette araştırırken bu niye çalışmıyor'a bakıyorsun, bunu nasıl çalıştırıyım'a bakmazsın. Yakın zamanda bir hata aldım, bu hatayı neden alıyorum diye yazdım. Bu hatayı alan biri var mı, benim gibi birisi bu hatayı almış mı, mesele bu. En çok baktığımız şey bu. [Fırat, 30][17]

In this regard, internet can be considered as a platform where specific kind of experience, i.e capacity to identify problems are transmitted. Coming up with detection and solution of problems becomes remarkably easier and saves workers' time and effort, accordingly. Thereby, they don't need to apply trial and error kind of methods which are generally tiring and time consuming.

Skill and knowledge renewal which will be discussed further in subsequent chapter is prevalent issue among software developers. As expressed in research of Burchell

[17] I type it every time and check out the first 5-6 pages. If they fit, I will use but if not, I close it down. You have to use this. Sometimes you come across with such annoying things. You check it over and over again. You don't understand why the code doesn't work. Once you really examine it, you find out what is wrong. This is where internet is most useful. While searching on the internet you search for the possible reasons of error rather than how to run it. Recently I have received an error and searched it on the internet. I looked up who else might have had the same error. This is the most common thing we look up. [Fırat, 30]

et al. (2002, 64), lifetime of skills and knowledge is getting shorter every day. It is the case especially in such industries where knowledge cultivation is the major drive of market competition. For most workers, therefore, professional development is a prevalent concern to be dealt with individually.

> Bu sektörde öyle, mesele maaş değil aslında. Ne kadar daha çok öğrenebilirim hastalığı var yazılım sektöründe. Bizde de var, bende var, ofiste çalışan arkadaşlarda da var. Herkes bir yandan web-development yaparken mobil teknolojilere bakıyorlar internetten falan. Halbuki mobil şeyimiz, müşterimiz yok. İphone geliştirmeye bakıyoruz hepimiz. [Emrah, 25][18]

Qualifications workers need to have have been rapidly changing I conjunction with the changing market conditions for companies and organizations. As a respond to these conditions, any practice which provides the opportunity to renew skills and knowledge is welcomed by workers. Internet as the medium which provides such activities seems to be the most effective and most commonly used instrument. Testimonies of developers which assert that internet is the first and most widely consulted instrument at their work prove its effectiveness compared to the traditional ways of learning.

Easily accessible character of internet stands as the first option to access knowledge necessary for software production. Davenport and Prusak (2000, 14) underlines the importance of accessibility as such: "The mere existence of knowledge somewhere in the organization is of little benefit; it becomes a valuable corporate asset only if it is accessible, and its value increases with the level of accessibility". Therefore, it is not surprising that knowledge on internet outcompetes the alternatives with regard

[18] In this sector, wage is not what matters. In this software sector, people are obsessed with how much more they can learn. I also have this, my friends also have this. Everyone searches for mobile technologies while doing web-development. We don't mobile thingy or client. All of us try to develop the iphone. This is a urge which forces you to learn everything. [Emrah, 25]

to accessibility and efficiency. As Cihan points out, compared even to recalling some practical knowledge used in his own previous work, searching it from internet is easier and preferable solution.

> Websiteleri çok kolay erişilebilir olduğu için her zaman daha kullanışlı oluyor. Kendi yazdığım koddan bir kısmı hatırlamam gerekse bile onu bulup anlayana kadar internetten bakması çok daha kolay oluyor. Öyle yapacağına yaz, string format yaz regular expression, zilyon tane şey çıkıyor yazdığında. [Cihan, 29][19]

As the quotes above illustrate, both its range and reachability make knowledge accumulated and produced on internet invincibly powerful vis-à-vis the knowledge residing in individuals' memories or completed works. There are plenty of developers who experience this situation in different places of world, Nate is one of them.

> I began as a developer before Google came out (1994) but since Google has become available I do not look back. I used to save all of my code for reference for future projects. It was a valued treasure at the time since I was building a useful arsenal. But because I can find much of what I am doing online I've realized that I no longer need to retain such knowledge long term. (Hanselman, 2013)

Therefore, individual developers' function as repository of knowledge within organizations has lost its viability to a considerable extent and this function has been replaced with online sources.

> İnternet, en çok o yani. Neredeyse her şey bulunabiliyor, özellikle yaygın kullanılan bir şeyse. Yeni teknolojiyse bazen bulamayabiliyorsun. Hiç döküman yok bir şey yok. Stack Overflow'da da yazan çıkmamış daha. Bir

[19] Websites are always more usable since it is easily accessable. Even I needed to recall something related with my previous work, checking it from internet would be more easier. Instead, type string format, type regular expression, search engines would bring millions of things. [Cihan, 29]

> şeyler deniyorsun, fikir verebilecek insanlara soruyorsun. Buradaki yöneticiler de ona yarıyor. Senin ne yaptığını bilmiyor, koda hakim değil, dile hakim değil ama kendi tecrübesiyle, bildiğiyle söylemeye çalışıyor sorunca. Aslında sorunca da değil de, noldu bizim iş diye geliyor da sen problemini anlatıyorsun. Abi böyle bir şeyle rastlaştım şu an, ondan ilerleyemiyorum diye. Şöyle şöyle yapabilirsin diye tavsiyelerde bulunabiliyor. Ama problem çözümümüz, sorudan morudan önce internetle oluyor. [Hakan, 26][20]

Dissemination of knowledge on internet has resulted in some differences in the software production, as well as in the role of developers. Emrah's brief comparison regarding the vitality of Stack Overflow [SO],which is a popular Question&Answer website for professional and enthusiast programmers, in production is exemplary in this sense.

> Stack Overflow [SO] olmadan kod geliştirilmez ki. SO yokken çok zormuş zaten. Şöyle, mesela benim çalıştığım şirketteki müdür o yıllardan gelme. 17 yıldır bu sektörde. Google'ın iş teklifi yaptığı bir adam, öyle birisi. Bir link'e 'href' vermek için arayıp bulman gerekiyordu bilen adamları, veya koskoca kitapları karıştırman gerekiyordu diyor. Ben ilk geldiğimde çok mızırdan-ıyordum, ya bu kod böyle yazılır mı, ne biçim yazılmış falan diye. 2004'de yazılmış bir de kod, söylenip duruyorum iş yerinde. Aynı yönetici o kod SO yokken yazıldı, ondan öyledir demişti bana. Öyle bir durum yani. [Emrah][21]

In those days, internet use was rare therefore knowledge was concentrated in the individual workers. Therefore, position held by the workers was more essential.

[20] Internet. You can find everything on the internet, especially mainstream things. But if it is a newly developed technology, you can't find any document, it is not even the "stack overflow" yet. you have to try new things, you have to ask people who might know. This is what managers do here. She doesn't know what you are doing; she doesn't know the code but she at least tries to explain things based on her experience. Honestly, managers explain things not when you ask but when they supervise things. During the supervision, you tell the problem you have come across and they give some advice. But first thing we consult in problem solving is internet. [Hakan, 26]

[21] Without Stack Overflow [SO], no code can be developed. It used to be extremely difficult before the SO. For example, manager of mine has been working in this sector for 17 years. Google has offered him a position. He tells us that to be able to assign 'href' to a link, he would have to look for someone who knew or look through the piles of books.

Organizations would have to look for employees who had information on specific tasks. Today, it is significantly different. Organizations do not need to rely completely on the knowledge developers have. Vast amount of required knowledge is stored and it is available on internet independent of individual workers. Therefore, production is achieved through the help of the internet; the developer simply needs to be capable of searching, understanding and manipulating the knowledge instead of having or recalling it.

Shortly, as knowledge is accumulated, stored and distributed in accessible platforms, reliance of organizations on knowledge embedded in people has diminished, not completely disappeared, though. Therefore, internet use gives some flexibility to the organizations in the reproduction of production thereby of employment relations, since labor of developers has become more standard and interchangeable as their reliance on the internet increases.

Furthermore, production gets faster and cheaper with the increasing use of internet. Knowledge is an inconsumable good. Once it is posted on internet, its cost of reproduction and distribution is close almost to zero. As far as the efficiency concerned, that means the time and effort every individual worker has to spend to obtain that knowledge can be saved. As a result, developers are not obliged to obtain knowledge as a result of time-intensive learning practices. They do not have to dwell on or fully understand what they will acquire from internet before using. Quick access to knowledge and skills of manipulating it would be sufficient for software development. In this way, labor-saving is achieved by shortening the time of production.

> İnternetten arayıp bulabileceğini farkedince, yani ne aradığını ne soracağını öğrenince kafan da ona adapte oluyor. Ne olduğunu bilmesen de almıyorsun.

> Çünkü çok fazla şey yapıyorsun. Farklı yerlerde farklı farklı işler yaptım. Hangi birini öğreneyim yani. [Hakan, 26][22]

It would be an exaggeration to draw such a conclusion that internet renders all the knowledge of software developers completely obsolete. It is not true; developers and their labor power are still needed to manipulate this particular knowledge. And they are still valuable assets; because capability of accessing and processing knowledge still depends on some qualifications such as higher education or investing time.

But it can be claimed that due to the incorporation of internet into production process, the content of the labor is deskilled in the sense Braverman recognizes. It is undeniable that with the development of internet, software production process has had the capacity to store vast amount of knowledge. Average knowledge and skill content of production is definitely much more excessive today. But according to Braverman (1998 :294-295), deskilling is not limited only to the development of accumulated knowledge. One should examine the labor content of production to say whether there is a deskilling or not. Therefore, the question must be whether content of labor tends toward averaging or toward polarization.

Considering the difference in productivity between individual-based and community-based knowledge production practices, it can be claimed that technologies of internet cause polarization of labor in favor of communities. Its labor content is getting widened at the expense of individuals' labor content. Superiority of communities' practices, defined as group-sourcing or peer production, over other alternatives of knowledge production and sharing, proves this

[22] Once you learn what to search on the internet, I mean, once you learn what you ask to the internet, you adapt it very easily. You don't have to memorize, comprehend it. Because you are expected to fulfill too many tasks within limited time. So, you generally have no time to memorize all of them. [Hakan, 26]

claim. Average of generalized knowledge and skills are increased, but majority of developers as individuals remain behind of it. Intellectuality of mass compounded by individuals is being incorporated into production, though developers confront with its negative consequences, which we will see below.

Recall that, in Emrah's example, thanks to the SO (Stack Overflow, Q&A website), majority of developers do not have difficulty to give a 'href' command to a link any longer. That is to say, the average time spent on the application of this practical knowledge has also decreased. What is really important here is that, the knowledge developer has and value of the time and effort an individual developer invests are equalized with this lower average. It can be said average brought by internet would always be lower because community based knowledge production and its transmission is unrivalled in terms of efficiency and productivity. That is to say, the need for the labor within organizations is reduced to some extent because significant part of work is fulfilled, that is, the information is produced, outside of the firms, on the internet.

This is what has changed: the effort that had to be spent individually to obtain knowledge through time-intensive learning practices is collectivized on the internet. This process of collectivization prevents the doubling of effort, thereby the labor. However, efficiency of such collectivization of effort does not necessarily protect individual contributors from the risks of being redundant or unemployed. Dependency of workers on income provided by market still maintains its significance. As Bauwen (2006) points out, "peer production on internet covers only a section of production, while the market provides for nearly all sections". What we witness here, according to Benner (2002, 8), is the growing individualization of employment in conjunction with increasing socialization of work. Therefore, the tension, as Thompson (2003, 364) notes "between what capital is seeking from

employees in the labor process and what it finds necessary to enforce in the realm of employment relations" still prevails.

In this respect, with the help of other factors, internet has changed and diminished the average requirements demanded from labor. Acquiring knowledge which was accumulated through working experience has started to lose its meaning. This has led us to the situation that value of retaining knowledge acquired within years has been depreciated, not fully though, but to some extent.

As a result of this transformation, majority of developers strongly emphasize that they need to develop some new skills to utilize internet.

> En büyük şey, yani kaynak internet. İnsanlar ilk olarak oraya bakıyor. Hakikaten ciddi bir know-how yatıyor orada. Nasıl arayacağını bilirsen de, mutlaka cevabını buluyorsun. [Yüksel, 40][23]

As Yüksel emphasizes, after internet, knowing how to search and how to learn constitute new skills which developers should have. These new skills, clearly do not refer to past achievements like possessing knowledge which is a result of time-intensive learning process and memorizing it.

As Sennett (2006, 15) points out, modern working culture "militates against the ideal of craftsmanship, that is, learning to do just one thing really well; such commitment can often prove economically destructive." Instead, potentially, more generic abilities are appreciated now because they are more reliable in a volatile market conditions which have been rapidly shifting and altering the knowledge and skills necessary for production.

[23] The biggest resource is Internet. People primarily look up in the internet. It reserves a critical know-how potential. If you know how to look up, you will definitely find the answer. [Yüksel, 40]

> Ben genelde şöyle öğreniyorum. Bir şeye ihtiyaç olduğunda sana geliyorsa talep, onu bir şekilde çözüyorsun. Benim öğrenme tarzım biraz öyle. Oturup manuel okuyan insanlara da hayranım. Zaman bulursam ben de okumaya çalışıyorum ama genelde bir kriz olduğunda, sorun olduğunda, bunu yapmamız gerek dendiğinde öğreniyorum. [Ezgi, 24][24]

It is observed that most of developers have adapted themselves to this transformation and have embraced these skills as much as they have embraced the internet use as a working practice. Rather than knowledge itself which is risky to invest in this volatile market, transferrable skills like knowing how to find and how to learn should be focused on in order to respond shifting demands (Thompson, 2003: 365). For example, according to some of developers I have interviewed, education of how to utilize internet for self-learning practices has to be the priority of higher education.

> Üniversite'de bu iki şeyi, neyi nasıl arayacağını ve neyi nasıl öğreneceğini çok iyi öğrenmen gerekiyor bence. [Erdem, 23] [25]

This situation also corresponds to Sennett's observation about the change of meritocracy idea among workers. As he points out, in new working culture, the measure of meritocracy has been shifted towards responding and solving problems rapidly instead of dwelling on it (Sennett, 2006, 15).

However, implications of change towards prioritizing potential abilities, which compel developers to switch projects very often, cause a tension among those who

[24] This is how I learn. If there is a demand, you have to find the solution. This is my style of learning. I admire people who read from the books. I also try to read but I usually learn while doing, facing with crises or problems, when supervisors told me that crisis or problem need to be solved immediately. [Ezgi, 24]

[25] You need to learn two things throughout the university education: firstly how to make research; secondly, how to learn. [Erdem, 23]

are working more routine and less skill required projects. Kutay's experience is helpful in this sense. He is heavily affected by the changing requirements and he faces with the afflicting ambiguity brought to the working experience of developers.

> Şöyle söyleyeyim, internet olmasa ben kod yazamam, o kadar. Unutuyorsun, çok ileri düzey şeyleri değil ama, unutuyorsun. Java'da nasıl array tanımlanır bakarım, utanmıyorum da, bakıyorum. Onu da araştırdığında biri sormuş oluyor, genelde SO'da sormuş oluyor. Çok dil değiştirmezsen gerek de kalmıyor aslında. aşırı bir örnek verdim ben. Bir sene boyunca PHP çalışmışım, Java'nın yüzüne bakmamışım, lazım olduğunda haliyle bakıyorsun. [Kutay, 25][26]

What he mentions is that he uses internet to find very fundamental information which every programmer is expected to have. However, his excuse is also valid, he had not been writing code with java for more than a year. The knowledge on PHP language might not be applicable in java and probably this is the case. So, who could blame him for checking that, if it doesn't affect his work negatively? Apparently no one, because, all young developers I have interviewed told similar examples about how basic and simple things they could look up on the internet as much as complex ones. Then what makes Kutay so uncomfortable as to propose excuses about his act?

The excessive reliance of developers' on the online sources appears to degrade work of developers and accordingly destroys their perceptions of work-identity. Keeping in mind that even copying available codes from other sources and applying it on your own work is also an available option, there seems to be little left creative or challenging to do for some developers. They don't have to put any effort in

[26] Let me put it this way: without internet I cannot write code. I forget, not the higher level things but many thing, I forget. I look up how to identify array on Java; I don't feel ashamed. When I look it up, I usually find out someone has already asked in on the SO. If I don't change languages very often, I won't even need. This is an extreme example, of course. But I have worked on PHP for a year and I haven't used Java even once, naturally, I check it out on the internet when I need to use. [Kutay, 25]

obtaining knowledge, if what is needed is already produced and ready at hand. Then what really makes a person a software developer? As the opportunities offered by internet and other production tools increase, it gets harder to find an soothing answer to a particular question, and self-confidence of developers and belief of them in their ingenuity is getting destroyed.

Popular blogger Scott Hanselman, a software developer, mentor and employee at Microsoft, touches upon a similar question in his blog. One of his followers sends him an email and asks for help to deal with tension he experienced. The question is “am I a really a developer or just a good googler?” (Hanselman, 2013).

The post receives hundreds of comments from developers stating how they have been moved by the question, because they frequently think about the same while using internet for work-related purposes.

In most of the comments, however, developers seem to oppose that internet directly degrades their working experience as developers. According to developers, it just replaced other source of knowledge like manuals, books and etc. Therefore, utilization of internet wouldn't be such a big problem if developers care to fully understand what they access from internet and respect their craft. Comment of Dennis Suppe (Hanselman, 2013), who has been working as a developer since long before internet, asserts that internet per se doesn't create googlers as a substitute of developers. Developers who slack to dwell on what they reach and use from internet are at blame for such degradation.

> "I've even seen developers who graduate with a CS degree and have trouble writing a program from scratch. They are so dependent on the Web for answers that they have trouble when asked to build something from the ground up.

> Don't get me wrong. I'm a fan of having access to information. But if you are a developer, you have an obligation to build quality code and to understand how it works. Using Google doesn't make you less than a real programmer. Not understanding your code and your craft does." (Hanselman, 2013)

Following this, Scott suggests some solutions to his followers to cope with this situation. One of them is not to use google or internet some days at work. However, is it really possible to do programming without using internet? According to Hakan, yes, you could do programming without internet, you could get used to it over time, but you would not because of the time constraints. As indicated in Sennett's analysis of new working culture, the pressure to speed up the work and to get results as quickly as possible are intense in new institutions and this pressure compels workers to skim on rather than to dwell (Sennett, 2006, 127).

> Kendini ona adapte ediyorsun zaten. Ben okuldayken hiç kullanmı- yordum interneti. çünkü çok daha belirliydi ödev formatları, bir takım inputlar vardı, outputlar vardı. Arada yaptığın şeyler de programlama dilinin kapsamı içinde şeylerdi. O yüzden hiç kullanmıyordum internet falan. Herşeyi öğreniyordum. Ama işe girdikten sonra öyle olamayacağını farkettim. Çünkü çok kısa sürede çok şey talep ediyorlar senden. Yeni yeni öğreniyor oluyorsun, bilmiyor olabiliyorsun. Başka başka teknolojiler oluyor falan. Nereye kadar öğreneceksin. İnternetten arayıp bulabileceğini farkedince, yani ne aradığını ne soracağını öğrenince kafan da ona adapte oluyor. Ne olduğunu bilmesen de almıyorsun, öğrenmeye uğraşmıyorsun. Çünkü çok fazla şey yapıyorsun. Farklı yerlerde farklı farklı işler yaptım, hangi birini öğreneyim yani… [Hakan, 26][27]

[27] You are easily get adapted to internet. I was not using the internet in the college. Because formats and instructions of assignments were much clearer; you could see the input and output. Works I was doing were within the limits of programming language. That's why I never needed the internet. But after I started working, I realized this cannot be sustained in the work. Because work demands too many things in a very short time. I have to learn from the beginning. I might have come across with a new technology. I can't memoriz all of them. Once you realize you can find everything on the internet, that is to say, once you learn which questions to ask, you adapt yourself to it. Because I have to tackle with too many things, different tasks in different places. How could I have learned all of them... [Hakan, 26]

In the same blog post mentioned above, as distinct from the others, comment of another visitor draws attention to the aspect of working conditions. Similar to Hakan, his statement shows us that time imperatives leave a little room to learn and fully understand knowledge they acquire from the internet.

> "The downside of that is most developers would just quickly focus on the next task after they've found a solution "that works". There is little time in today's work reality to really sit down to read and experiment to gather the deep understanding that is actually required to solve those problems. The law of project schedules dictates you have less time to finish the work than the time required to learn the technologies proficiently to do the work in the first place.
>
> This is a work culture that needs to be solved by both developers and managers/bosses; especially the latter who almost never allocate enough time for tired staff to catch their breath." (Hanselman, 2013)

Therefore, using internet is rather an obligation than a choice in most of the time. Hollowing out of abilities of software developers via internet, thus, is connected to organizations' tendency to fasten production and achieve labor savings.

It should be pointed out, however utilization of internet does not serve the same purpose for every software developers. According to the skill requirements of project being worked on, relation with internet use may vary between developers.

> Çok çok kullanıyorum Stackover Flow'u [SO], Github'ı. Benim karşılaştığım sorunlar birebir kesinlikle yok. Ama konfigürasyonla ilgili bir sorun olduğunda direk buluyorsun. Bir tane algoritma var mesela, Wikipedia'da bile yazıyor, nasıl çalıştığı, objenin köşelerini buldurmaya yönelik bir algoritma. Bunun hazır kodunu alıp direk koyamıyorum, çünkü senin mimarine uygun olanı, efficient çalışanı yazman lazım. Kendim bir şey yazmak zorundayım hep. [Ceren, 25][28]

[28] I use the SO and Github (social code repository website) very frequently. I can never find exactly the same problems I face. But once I come across with a problem about configuration, I immediately find it. For example, there is an algorithm, the instructions to

As the lines stated above, there are still critical to apply practical knowledge, that is, the know-how that has not been circulated on the internet yet. Therefore, required labor power has to be skilled enough to fulfill those tasks. Ceren works in this kind of exclusive project whose know-how is not available on internet. Therefore, it is difficult for her to find and use solutions on the internet. I asked her whether not using internet as much as other developers do is a problem for her or not. Her answer was not definitely discontented. Lack of distributed knowledge over internet as well as lack of resources like production tools seems to please her contrary to what is thought.

> Artık yazılım işi biraz abstract olmuş. Fiziği bilmek zorunda değiller artık. Sadece çağırıyorsun, mesela destroy object diyor, class'ını yazıp .destroy diyorsun, destroy ediyor, bu kadar. Adam nasıl destroy ettiğini bilmiyor. Benim işim bütün bunları bilmemi gerektiriyor, performansa yönelik bir iş yaptığım için. Bir yandan iyi çok iyi, ben ileride Nvidia'da çalışabilirim ama onlar çalışamaz. [Ceren, 25] [29]

In this regard, it is striking to see that internet also helps sharpening the technical division of labor among the employees with regard to criticalness of the task. It helps as Braverman (1998: 57-58) asserts, to classify workers as those "whose time is infinitely valuable and whose time is worth almost nothing". Then, it would not

---

use, to run it are written even on Wikipedia. But I cannot apply the same code which is provided because I have to find the most efficient one in accordance with my own project. I always have to write it by myself. [Ceren, 25]

[29] My work is “pure software development” I guess. The others usually do abstract works. They use pre-made functions, enter parameters. They don’t know how it works, what is the algorithm, how is the performance. Today, software development has become more abstract. You just call the function. No one has to know about the physics operating behind it. For example, she gives the command “destroy object”, she writes the command and it destroys itself. That’s it. She doesn’t know how this function destroys. In my job, I need to know all of these things, because performance of software application is very critical. On the one hand, this is very valuable for me. In the future I can work for Nvidia, but the others can’t. [Ceren, 25]

be coincidence to hear from Ceren that the project, she is currently working on, will be beneficiary for her career since projects' know-how can not be found on internet.

Compared to other instruments of production I have briefly discussed in previous sections, internet assumes a more critical position in the process of software development. This difference stems from its ability to produce and disseminate knowledge as well as its tremendous speed in the sense of response time to the needs of production. In our interviews, knowledge production and sharing practices are praised by developers from a voluntaristic and cooperative perspective. However, as we have seen, incorporation of these practices within capitalist mode of production has bolstered domination of capital on labor. It has indirectly strengthened the subordination of workers to capital by decreasing their control over production process.

So far, I have discussed how advancement of production instruments and specific technical organization of software development have been shaped by and served to the interests of capital. However, practical use of these instruments, as well as translation of labor power into labor is distinctive issues which we can only observe through relations of control between employees and employers at workplace. Because, technical organization of production and its targets over production process cannot be carried into effect own by themselves, they have to be implemented within specific settings. In final stage, therefore, agency of workers either by way of coercion or consent is inevitably required.

CHAPTER III

RELATIONS OF CONTROL OVER PRODUCTION PROCESS AT WORKPLACE

No matter how means of production are determinative on production organization, implementation of them and production of surplus value is ultimately actualized at the workplace, within the relations between workers and employers. What we are trying to say is, if a worker does not implement programming paradigms or production technologies in a proper way or totally refuses to use it, capitalist may not accomplish the surplus value production, hence cannot make profit at the end. Therefore, as well as the technical organization of production, relations in workplace are crucial part of production process, hence should be evaluated to comprehend characteristics of software development process in Turkey. In this regard, we are going to inspect relations of control at workplace between software developers and managers.

To begin with, it should be stated that what capitalists buy from workers is not the exact, concrete amount of labor at the beginning, but the potential of it. Thus, the degree of realization of this potential as concrete labor shows uncertain feature.

In capitalist production, this degree of realization determines the rate of profitability, more precisely the rate of surplus value production what capitalists and managers strive for. As we know from Marx (1992, 644), capitalist production is the production of surplus-value in the first place rather than production of commodities. Therefore, the translation of labor power into labor has to be handled under the

control of capitalist in a way to enhance surplus value production. Capitalist control over production for this reason is of great importance for capital.

In the sense of surplus-value production, there can be mentioned of two ways of raising rate of surplus value. First one is the production of absolute surplus value based upon the prolongation of the working-day beyond the point at which the labourer would have produced just an equivalent for the value of his labour-power. And the latter is the production of relative surplus value based upon the raise in productivity of labor power, in which the necessary labour is shortened thereby equivalent for the wages is produced in less time.

However, as asserted by Braverman (1998: 39), the relation between capital and labor in the sense of exchange-value, which is decisive on the rate of surplus value, is antagonistic one due to their irreconcilable interests. That means employers' and employees' interests constitute a potential source of conflict in any attempt to determine the utilization of labour power. Emrah's statement about the tension between employee and employers supports Braverman's account of conflictual interests:

> Çalışanla işveren arasında şöyle bir ilişki var. Bir tanesi ne kadar az çalışacak diye bakıyor, patron hep şüpheli yaklaşıyor, ne kadar çok kaytaracak acaba diye. Diğeri de [çalışan] beni aldığım maaştan daha fazla çalıştıracak mı diye bakıyor. Patron kaytarmayı minimuma çekmeye çalışıyor, diğeri maaş performans oranını maksimumda tutmaya çalışıyor.[Emrah, 25][30]

[30] There is such a relationship between the employee and the employer: one [the employee] always seeks for working less, the other [the boss] is always suspicious trying to catch the other shirking her duty. Employee, on the other hand, tries to grasp if the employer will assign more workload than she pays. The employer tries to minimize the shirking; while the employer tries to maximize wage-performance balance. [Emrah, 25]

Management, for this very reason according to Braverman (1998: 68), should essentially be located at the top of production process and execute it accordance with the capital's interest. They have to consistently regulate and maintain production organization such a way that surplus value can be produced and appropriated.

However, Braverman's account of control, especially its tendency to objectify dynamics of capitalism as the main drive of labor process and control formation is criticized, since it has been found too simplistic and unilateral by his successors. According to them, control, in Braverman's account operates without any complication, or interference of workers in the direction of dynamics immanent to capitalist mode of production, particularly to the search of surplus value extraction and accumulation of it. His emphasis especially on scientific management based on Tayloristic principles is criticized to be overwhelmingly deterministic and functionalist since resistance of workers and its effects on formation of control are largely neglected. For Burawoy (1982: 15), even frictional effects of workers are determinative rather than derivative, in the production process thereby in the formation of control relations. That is to say, resistance of workers or absence of it has as much impact over production process and formation of production relations as tendencies of capital to revolutionize production process and means of control it.

In this manner, Burawoy opposes to Braverman's claim that control is derived and based upon the irreconcilable nature of interests of capital and labor. According to Burawoy, Braverman grounds his notion of control upon a necessity based on antagonistic relations between worker and capitalist. This approach would be deficient and misleading since such conflictual foundations could not account for the prevalence of control practices based on cooperation and consent in workplaces (Burawoy, 1985: 24). In regard to exchange value, he agrees with Braverman; interests of capital can only be accomplished at the expense of labor. This is a

potential source of conflict between these two. However, as Burawoy (1982: 32-35) proposes, by obscuring surplus value extraction and offering concessions to individual workers, interests of workers and capitalists would be coordinated in such a way that subordination of workers can be achieved on cooperative basis. One of my interviewees', Emrah's portrayal about the relation which ought to be between employee and employer might be exemplary to reconciliation of interests:

> Şöyle; patronun kafasında aslında şu var. Tek derdi para kazanmak. Kimse kimseye karşılıksız bir şey yaptırmıyor. Kazandırdıkça şirketin bir parçası haline geliyorsun çalışan olarak. Sen kazandırdıkça o seni daha fazla motive ve mutlu ediyor. Fakat bunu hissettirmen lazım, çalışanın bunu bilmesi lazım. Yazılımcının az bulunduğu bir sektörde düşün ki çalışan yazılımcı bütün işlerini ayağa kaldırıyor. Ona yaklaşımın nasıl olur, ortak bile edersin firmaya, çünkü gitmemesi gerekiyor, birisinin daha iyi bir teklifte bulunmaması gerekiyor, hisse vereceksin, ortak yapacaksın, anlatabiliyor muyum. [Emrah, 25][31]

Kutay's experience can also be taken as an example of how prolongation of working day may be achieved on cooperative basis. For Kutay, being acknowledged and appraised by employers for example might be notable reason in exceptional cases to work willingly for long hours.

> Bir insanın takdir edilmesi gerekiyor bence. Bu hani o kadar çok değiştiriyor ki. Şu son ofisler ayrıldığından beri, öteki ofiste iki ayda yapamayacağım kadar iş yaptım. Niye ? Çünkü patron proje yönetici gibi bir pozisyonda. onun yanındayım, takıldığım yerde soruyorum, bunu böyle mi yapalım böyle mi yapalım diyorum. ya da güzel bir şey yapıyorum, 'bak böyle bir şey yaptım' diyorum. 'İyi güzel yapmışsın, baya söktün sen bu işi' diye

[31] The employer always and only thinks making money. No one asks anyone to do anything for free. As you bring in, you become a part of the company. As you make more money or the company, this makes you happier and more motivated. However, you need to make the employee understand this; the employer needs to know this. Think about this, in a sector where there is a few software developers, a software developer tackles with all the works, steps up. How do you approach her? You even make her a partner of your firm. Because she needs to stay; she must not be offered a better position by someone else. You will keep with you, you will give her share in the company. Do you know what I mean. [Emrah, 25]

> veriyor gazı arada. daha istekli çalışıyorsun. Yalan olsa bile takdir edilmek seni motive ediyor. Tersi olduğunda da, 'bu niye bitmedi, şunu şu zamana bitir, çok geride kalmışız oldu mu' dendiğinde de, daha isteksiz çalışıyorsun. Ama bir kere şey oldu, patronun olduğu ofise geçtiğimde, yarın sunum yapacaz, bu sunum önemli, şunları tamamlayalım denildi. Patron da oturdu, benle birlikte çalıştı. Bir gün sabah 9 10 gibi başladım, yetişsin diye ertesi gün akşam 7'ye kadar çalıştım. [Kutay, 25][32]

Therefore, worker's subjectivity is also determinative aspect of formation of control relations. That is to say, their resistance or consent cannot be disregarded within the relations of control. Thus, control strategies of management aiming to increase surplus value production by cultivating consent among developers should be evaluated as well as the strategies of management based on coercion.

In this regard, Friedman (1977) proposes a dichotomy between direct control and responsible autonomy as a base point in order to discuss the relations of control. Direct control refers to the strategies of management imposing direct constraints over freedom and discretion of workers regarding the immediate production in his account.

However, there are some cases that direct control strategies might be counter-productive like where cooperation of creative and skilled labor are indispensable part of production. In these cases, "workers could be subject to forms of direct control, but, there were limitations to it" (Kitay, 1997: 2). If these limits were

[32] I believe, one needs to be appreciated. This makes such a big difference. Since I left the previous office, I have done twice as much work as I can do in the previous one. Why? Because, the boss works as a project manager. I am with her. I ask when I need to ask: should we do this, should we do that? Or I do something new and nice and tell her. She says "good for you, you have learned this job". She encourages you. Even though this is a lie, you get motivated by being appreciated. Otherwise, when the boss puts pressure on you by asking "why this is not finished yet?", "submit this on time", "we are falling behing the schedule", then you work unenthusiastically. Sometimes such thing happen: the boss says "we have a presentation tomorrow, an important one. Let's finish these." She works next to me. One day, I have started working a 9-10 a.m., I have worked until 7 p.m. the next day. Just to catch up the work. [Kutay, 25]

harshly imposed, workers may start to show resistance against management. Even they might start to consider quitting their job for another.

For such situations, bestowing autonomy including authorization or responsibility which workers would think that they have the freedom to take decisions related to work are put forward as a managerial strategy of control. Friedman (1977) grounds his notion of responsible autonomy based upon these strategies of control that strives to cultivate consent of workers. By giving autonomy to the workers, management aims to stimulate workers' own sense of self-discipline rather than giving a freedom related to work (Barrett, 2004, 785). Hence, managerial targets such as increase of surplus value production or workers' subordination to management can be achieved in a frictionless and more applicable way.

Responsible autonomy concept as a strategy of managerial control can be evaluated under two subsets as technical autonomy and time autonomy. Technical autonomy refers to the power and value which high-skilled workers can derive from technical skills and profound knowledge they have over production (Scarbrough, 1999, 11). Benefits of such strategy might be rich for management, because it can be used as workers decide the best method to achieve the desired outcome (Barrett, 2004, 786) especially where managers are not knowledgable enough to decide which method is the most efficient and productive one.

Time autonomy, on the other hand, is a strategy of management to achieve best outcome by granting some degree of control to workers over management of working time. However, that does not mean workers would have full discretion on their working time arrangement. They are still obliged to manage their time within the context of project deadlines. Generally, these deadlines of tasks or projects are designated by managers, but responsibility of time management within these timeframes and finishing assigned tasks is left to the workers. With this way,

employee commitment and consent is desired to be achieved through practices that feed and serve the demand for autonomy and by creating a work environment in which individual development is fostered and encouraged (Voss-Dahm, 2001). In this section, following the brief theoretical trajectory above, findings coming from field study about relations of control are going to be discussed. Through which ways control relations have been formed are also explored.

3.1 Relations of Control over Technical Aspects of Software Development

In the matter of the discretion over the way of execution of production, managers' and workers' interests rarely coincide with each other. Developers want to be free about the way of doing their job, while managers want to ensure shortening production process and making a profit at the end of it. Therefore, it is not surprising to encounter with cases which managers' decisions in the form of direct control contradict with developers' demands of discretion and autonomy over technical aspects of production. Experiences of my respondents when managers force developers to do programming in a particular way that they choose to do, and their responses to it are exemplary in this sense:

> Özetle şöyle bir şey var, bu yazılım sektörüne özgü bir şey olabilir, özgür olamıyorsun. Yazdığın koda kadar karışıyorlar, gelip burayı öyle yapma şöyle yap diyebiliyorlar. Aynı işi yapan kod bile olsa karışabiliyorlar. Adam geliyor mesela, bunu böyle çöz diyor. Öyle çözmek benim aklımda yok, o kendi kafasındaki kare çözümü benim daire çözümüme oturtmaya çalışıyor. Özgür hissetmiyorum, her alanda dizginlenmiş hissediyorum. Yazdığım her satırda, lan ben bunu böyle yazıyorum ama bu adam bunu şöyle düzelttirecek, ben en iyisi böyle yazıyım diyorum. Bu beni kısıtlıyor. [Leyla, 25] [33]

[33] Long story short, there is such a thing: this might pertain to software sector – you can't be free. They interfere even with the code. They can say "don't do it like this, do that like this". They interfere even though the code works as the same way. Not because there is a regulation, but because the guy in charge says so. There is a pattern, a coding style, this is not about it. The boss comes and asks you to decode something in a way she asks you to do.

> Bir iş görüşmesine gitmiştim, baya saçma bir insandı görüştüğüm yönetici. Ofiste biri çalışıyor, bizim olduğumuz yerden de gözüküyordu. Elini yüzüne koyuyormuş çalışırken. Onu gösterip, öyle çalışılmaz ki, şöyle çalışılmalı böyle çalışılmalı falan diye anlatmıştı. abi işin içine çok giren adamlar hep böyle oluyor anladığım kadarıyla. robot gibi eleman istiyorlar. hiç sıkıntısı olmasın, hangi ortama koyarsam çalışsın en yüksek performansla. [Kutay, 25][34]

Managerial interferences are rarely welcomed by developers. They don't want to see to have the opportunity to make decisions about their work taken away and given to someone else. Developers, in these circumstances, are apt to get unhappy and resentful. They are prone to show resistance against efforts to reduce them to extensions of machines someone else has designed (Kraft, 1977, 20). Therefore, direct control may not overlap with managerial targets like enhancement of productivity and profitability especially where the cooperation of skilled labor is proved out to be more productive. For this reason, direct control may turn out to be counter productive.

As indicated by Leyla, imposition of technical instructions of managers may slow down developers' pace and tend to simplify their work. As work and its way of

---

However, I don't plan to decode that way. She imposes the decoding she has in mind. Therefore, I get slow down, work gets slow down, I don't like the work anymore, so and so forth. I don't feel free; I feel curbed in every sense. Every code I write, I have to keep that those concerns in mind. I think the boss will change this, she will ask me to do this in another way. This constraints me. [Leyla, 25]

[34] I have once been to a job interview at the company where one of my friends used to work. The person I met, she was awkward. Someone was working in the office and from where we were sitting, we could see her. She puts her hand on her face while working. This person I had a meeting, she showed her to me and said "this is not the true way of working. One should work like this, like that". To the best of my understanding, those who are heavily into the job are like this. They want robot-like employees. She shouldn't have any problem, she should have high performance every time and any time. This is awkward. [Kutay, 25]

execution become less challenging, developers' creativity and self-developmental targets are more likely to be interrupted. Thereby, problems and confrontations between workers and management emerge. Because, when work is simplified and routinized, to keep skills and knowledge up to date is getting difficult for developers. As a result, developers which are mostly identified with and committed to their content of work rather than the organization (Barrett, 2005, 83) consider leaving the company for another which they think they would develop their skills as Leyla did.

What should be stressed here that management could have increased profitability by exercising direct control and routinizing the work, but employees would have left the company. As proposed by Kitay (1997, 2), forms of direct control could be imposed upon workers, but, there were constraints to it. If these limitations were overcompelled, direct control would start to disrupt production process on the contrary of targets and intention of managers. However, managers and employers generally do not want to lose their employees, especially in the case they are not easily replaceable for sustaining production without loss of time. Therefore, they try to keep balance between direct control and autonomy.

3.2 Relations of Control over Working Time

Bestowing technical autonomy to workers for these reasons appears to be more applicable/viable strategy for management as workers decide the 'best' method to achieve the desired outcome with commitment to their work instead of showing resistance, on the condition that interests of capital, accumulation of surplus value are secured by other means of control (Barrett, 2004, 786).

Having strict control over determination of time period which projects have to be completed is one of those. In this regard, arrangement of project deadlines can be

asserted as the most commonly utilitized means of direct control in Turkey. Instead of dealing with performance of developers which is still hard to measure and monitor, managers exercise control through certain quantity of product by imposing project deadlines.

> Türkiye'de bunu yaptığını iddia edenler var, ama ben hepsine şerh koyarım. Kimse yapamıyor bunu. Ben de tam yapamıyordum, söylüyüm. Niye biliyormusun? Yazılımın doğasından kaynaklı bir durum. İnsanla yapılan bir şey yazılım. Her zaman unpredictable şeyler çıkabiliyor ortaya. Yoksa bir yazılımcının eforunun yeterli veya yetersiz olduğunu değerlendiren bir sistem olduğunu düşünmüyorum. O yüzden projenin maliyeti ve zamanı üzerinden hesaplanıyordur. [Cemil, 34][35]

As also pointed out by Cemil, hardly measurable characteristics of developers' work performance, hence uncertainty impels management to demand specific amount of concrete labor from workers for specific amount of time. With this way, straining intensity of developers' labor power can be remotely achieved by exercising deadlines.

However, workers' concern of their leisure time and employers' or managers' demands to put workers' time to productive use generally contradict with each other (Voss-Dahm, 2005, 109). Accordingly, problems and disputes between management and workers are most visible when setting of project deadlines begins to compel workers to do overtime. It is not surprising then, intense workload and overtime is one of the most prevalent problems of developers in Turkey. As stated by my

[35] While you are dealing with the software, you also need to deal with it at the technical level. Beside this, you have to conduct things at the effort, time and communication level. In Turkey, there are people who claim to achieve this. But I doubt that. No one can do this. I can't do this properly, I must tell. Do you know why? This is because of the nature of the software. Software is a human made thing. There is always unpredictable things to come up. That is why plans always are made upon cost and time of software projects. [Cemil, 34]

respondents, working for long hours appears in Turkey as a very ordinary demand of management which developers have to response by any means.

> Bu şu anda part time çalıştığım şirketi de sayarsam, bu 4. şirketim, en kurumsalında da çalıştım. Ama yok yani, arkadaşlar 2 ay boyunca akşam 9'da çıkılacak diye mail gelebiliyor. Hiç kimse de fazla mesai parası almıyor falan. Bunlar hoş şeyler değil, şirketlere özel şeyler değil. Bütün şirketlerde bu böyle kanıksanmış, böyle kabul edilmiş. İnsanlar şey demiyor hiç bir zaman, ya 3 gündür saat 8'de çıkıyorum arkadaş, burada benim hakkım yeniyor demiyor. Çünkü bir bilgisayar mühendisinin üst üste 3 gün saat 8'de işten çıkması gayet normal bir şey olarak algılanıyor. [Leyla, 25][36]

It is expected from software developers in Turkey to accept overtime or intense workload as a natural outcome of software development. Related with it, they are supposed to do overtime to compensate any latency or complication related with conduct of production, irrelevant from whether they are responsible from those problems or not. Similar to what Leyla tells, Hakan word's verifies that doing overtime is legitimate demand of management in most of the workplace.

> Çok mesai yapılan yerlerde ücretlendirildiği oluyormuş, bankalarda mesela. Bu tip yerlerde de olabiliyormuş ya, şansıma denk gelmedi. Ama onun dışında çok doğal bir taleptir ek mesai, yazılımın doğası gereği, ya da bize öyle inandırıyorlar, ona da emin değilim. Burada hiç yapmıyoruz ama 3 gün sonraya bir şey yetişecektir. Yönetici gelir arkadaşlar 3 gün sonraya bir şey yetişecek der. Bu bir olgu gibidir, bir şekilde hep beraber yapıcaz deyip yapılır.[Hakan, 26][37]

[36] If I include the company I currently work part time, this is the 4th company I have worked. I worked even in the most corporate ones. But nothing is different. One day you can receive an email: "Dear friends, for a two-month period, you are going to get off work at 9 p.m." No one gets paid for overwork. These are not nice things, these are not peculiar to specific companies. This is how it is taken for granted in all the companies. Workers don't say "For three days I have been getting off the work at 8 p.m., this is violation of my rights." Because for a computer engineer, finishing up the work at 8 p.m. is perceived as normal. [Leyla, 25]

[37] In places where overwork is a common practice, like banks or places like this, people might be getting paid for overwork; unluckily, I have never come across. Apart from that,

As Hakan indicates, in very few workplace, overtime work of workers are paid by companies. In addition, workers seem to reluctant to request to be paid for overtime work. Thus, it can be claimed that absolute surplus value production through prolongation of workday in Turkey is a widespread way of making profit.

Many interviewees of mine state that, inadequate managers which fail to accomplish better negotiations, contracts with clients and to organize and design project properly is the main reason of overtime and intense workload they have to carry out.

> Yöneticiler de çok mühendislikten, işin arka tarafını, mutfağını bilmiyorlar ama bilen adam da sana aynı davranıyor. Yapacaksın diyor, bunu yapacaksın, şöyle olur böyle olur yapacaksın diyor. [Leyla, 25][38]

> Adam gibi projeyi yönetmek, müşteriyi yönetmek yerine, haydi yazılımcılar, haydi ameleler, çalışın diyebiliyorlar. Şimdi biz ayrıldıktan sonra haftada 4 akşam fazla mesai yapıyorlar, cumartesi çalışıyorlar. pazar günü de evden çalışın deniliyormuş. Bunlar insani koşullar değil. Ama bu baya piyasanın normali, o yüzden yöneticiler bunu yaparken de gocunmuyorlar. [Elif, 37][39]

However, as we spoke, it was revealed that poor management or ineligible managers who lack technical expert can not account single-handedly for short deadlines and concomitantly intensification of work. When I asked the reasons why

---

overwork, by software's very nature, is a very natural demand. Or they make us believe so, I am not sure. We don't do it here but there are such situations: there is a work to catch up in three days. The manager comes and says "dear friends, this job will finish in 3 days". This is almost like a fact. You do it altogether and that work finishes. [Hakan, 26]

[38] Managers usually don't know the engineering, they don't understand how it works. But even those who do understand do the same thing. They treat you in the same way. She says "you will do this! One way or the other, you will do this." [Leyla, 25]

[39] These are inhuman conditions. This sector is very suitable for this. Instead of managing the project properly, they put the burden on workers' [software developers'] shoulders. This the "normal" in the market, that's why employers, managers don't have a problem with doing this. [Elif, 37]

also managers who have technical background do not tell their superiors or clients that deadlines are not bearable according to hardship of workload developers have to undertake, Leyla and Yüksel answered as:

> Şirket sahibine kadar böyle bir yaklaşımın olması lazım. Genelde öyle olmuyor ama. O insanların da şöyle bir durumları var, yöneticilerin de üstüne yük bindiriliyor, onlar da yükselmenin peşindeler, bir adım daha ileri gitmenin peşindeler, onlar da muhtemelen bu yollardan geçerek geldiler. Onca sene çalıştık, biraz da biz rahat edelim kafası da var. Öyle olunca bu tarafı gayet unutup, kendi kariyerlerine, kendi yükselme durumlarına bakabiliyorlar. [Leyla, 25][40]

> Ticari mantık, yönetim kademesine yükseldikçe insanlarda niyeyse bir hortluyor. Bana şey gibi geliyor. insanlar senelerce siyaseten karşı tarafta yer alıyor, muhalefette kalmayı tercih ediyor ama iktidara geldiğinde bir anda değişiyor. O koltuğa kendisi geldiğinde farklı endişeler içerisine düşüyor, bambaşka yönetmeye başlıyor. [Yüksel, 40][41]

As indicated by Yüksel and Leyla, managers' individual interests are also effective drive of implementation of harsh deadlines. Managers have to look out for interest of capital to protect their position and to get promoted to upper positions of management.

[40] Usually this is how it happens: there might be some managers, engineers who insistently tell that the project does not finish by the time client asks. However, when their superior does not agree, the same thing occurs. From the engineer to the company owner, everyone needs to embrace this approach. Yet this hardly happens. Managers are also in different situations: they are imposed too much burden. They want to be promoted, they want to go one step further. They have probably been through all these things. They think "we have worked for years and years, now, let me have some free and relaxed time". Hence, they can easily forget what we are going through and care about their own position, their own promotion. [Leyla, 25]

[41] For some reason, once the people get promoted higher and higher, this commercial logic revives. It seems to me like this: people have been taking part in the opposition for years, they prefer to stay in the opposition but once they come to power, they immediately change. They start being concerned about different things, they run the country very differently. [Yüksel, 40]

In interviews, I have heard complaints couple of times from developers about increasing power of clients over production process. Clients' needs and wish lists appear to be turned into a means of direct control as they become the parameters within which the software product is developed (Barrett, 2005, 75). To respond demands of these "overhasty" clients, the ones who have to stay overwork and being faced with intense workload are again developers.

> Benim gördüğüm şu, müşteri her durumda haklıdır, müşterinin her istediği yapılmalıdır. Teknoloji çok ileri ya, adam böyle önünde açıyor, atıyorum chrome'u açıyor ve sürekli browser görüyor ya önünde. Ya arkadaş biz bunu yaparız, bunu yapın, iki tane buton koyacaklar oraya diyor. Bizim başımızdaki adam da, tamam ya elbet biz yaparız diyor, arkada bizi mesaiye kaldırabileceğini biliyor. Tamam yaparız, biz bir aya hallettik o browser'ı, işte o ürünü diyor. Gelip bizim tepemize iniyor. Yapacaksın diyor, bunu yapacaksın, şöyle olur böyle olur yapacaksın diyor. [Leyla, 25][42]

Increasing effectiveness of clients' participation into production process renders developers vulnerable and helpless against clients' needs and wishes. As also indicated in Deery's research published in 2002, client's demands have to be met immediately as if they are coming from employers or bosses (cited in Yücesan-Özdemir, 2014, 156). Therefore, it can be concluded that imposition of client's needs and wishes in Turkey can also become instrument of absolute surplus value production.

[42] This is what I see: client is always right, whatever client demands should be met. Since the technology is advanced, the client comes and asks something. For example, she opens the Chrome, and since she sees a stable browser, she wants the same thing. She thinks this is just two buttons. And the guy, the manager, says "Ok! Of course we can do it. We can finish this browser/product in one month." Because he knows he can forces us to work overtime. He compels us. We confront with uninformed customer. I have never seen a customer who knows the production processes. Nor do the managers know how it goes. Even those who do know treat the employee in the same way. She says "you will do it! One way or the other, you are going to do this." [Leyla, 25]

After extracting and concentrating generalized knowledge of production in specific group of people, using this monopoly against workers one of the principles which Braverman has described how scientific management operates at production process. According to him, scientific management firstly gathers and develops knowledge of production by separating conception from execution of production, secondly makes it exclusive assets of itself and use this monopoly over the knowledge to control each step of production process and its execution.

Strategy followed by managers at workplace which Ceren is working share similarities with these principles of scientific management given above. According to Ceren's quote, managers of her company keep information about deadlines of projects hidden and don't share them with developers.

> Ve yazılımda şöyle bir şey var, derste de aldım, hocamız anlatmıştı. Projelerin ne zaman biteceği tam anlamıyla yazılımcıya söylenmez. Çünkü söylenirse o hep sarkıtır. Hemen bitecekmiş gibi istiyorlar zaten. Sürekli öyle konuşurlar, hepsi öyle konuşuyor. Ben geldiğimden beri bitecek de bitecek. Ne zaman bitecek bilmiyoruz. Ama var bir tarih, onların bildiği bir tarih var. Bu ödevin due date'ini bilince son güne bırakmak oluyor. Bilmemek en iyisi. [Ceren, 25][43]

Performance of developers is thought to be influenced negatively when they are informed about the project deadlines since they have a habit of procrastination like students postponing to study to the edge of exam period or submission of assignments. Similarly, developers may slack at work if there is a time they think they would complete the tasks or projects assigned to them. Therefore, gathering and hiding information related with deadlines and using it against workers aims to

[43] And in this sector there is such a practice: software developer does not know when the projects will end. Because if she knows, she would always delay it. The employers want you to finish it as if it will be submitted the other day. They always talk like this. Since I came here, it is going to finish. We don't know when to finish. There is a deadline they know. If I know the deadline, I will now do it until the very last day. It is better not to know. [Ceren, 25]

keep pressure of deadlines upon workers alive all the time. Management could more efficiently put developers' time in productive use with this way, and could make a profit by cheapening necessary labor.

We have observed that, in addition to direct control strategies, autonomous time management is also implemented as a control strategy in Turkey. Time autonomy means that developers are free to organize their working time as long as their tasks are completed till given deadlines. That implies deadlines act as a form of direct control, whereas employees have freedom to manage their time within the context of deadlines (Rasmussen and Johansen, 2005, 96). The middle size company which Emrah is working provides time autonomy to its developers:

> Çalışma saatleri 12-9. Bize her zaman ara zaten. Home ofis, yemeği de orada yiyoruz. Mutfağımız var. Ara diye bir şey yok bizde, şu zaman ara diye bir şey yok. Yemeğini ye, dışarı çık gel sorun değil, ofiste işin yoksa hiç gelme bile. deadline'ı yetiştir yeter. [Emrah, 25][44]

> Yok, mesainin tamamında çalışmıyorum. Öyle olmuyor, çok kaytarıyorum. Yemek oluyor, Türkiye'yi kurtarıyoruz, futbol konuşuyoruz. Yemekler zaten ofiste yeniliyor. Biraz facebook, youtube, onu ona gönderiyorsun, bunu gördün mü, o sana bunu gönderiyor, gördün mü diye. Tam yoğunlaştığın arada mesai bitiyor. [Emrah, 25][45]

Autonomous time management as a strategy of managerial control does not enforce strict working hours. Instead it promises that workers may have the freedom to organize and to manage their working time as long as they voluntarily took

[44] Working hours 12-9. But we are always in the breaks. Home office, we have our meals there. We have a kitchen here. There is no such thing as break; there is no pre-determined break. Have your lunch, go out, these are not problems as long as you catch up the deadline. [Emrah, 25]

[45] No, I don't work all time long during the work hours. I procrastinate a lot. We have lunch, we save the country, discuss football. We have lunch at the office. Facebook, YouTube, you send those to the others, "have you seen this?" At the moment you focus on working, working hours end. [Emrah, 25]

responsibilities of project delivery. Thus, autonomy and freedom at work constructs working hours including overtime as self-determined and their amount of work as something that they, individually, are responsible for (Rasmussen and Johansen, 2005: 101). As put it by them (2005: 105), "having taken on the responsibility to deliver on time, they are committed to meeting these deadlines". After he sorts out advantages of time autonomy he got, Emrah reveals that distinction between work and his daily life is getting blurred because of the time autonomy he has.

> Sıkıştığımız oluyor, haftasonu çalıştığımız oluyor. Çalışmasak oluyor ama kendimizi rahatlatmak için çalışıyoruz. Sürede sıkıntı yok ama işe gitme süresinde sıkıntı yok yani. İlla 12'de orada olacaksın diye bir şey yok, ben 1-1buçuk gibi gidiyorum ama 9'da çıkıyorum. Ama evden devam ediyorum, bazen hiç gitmiyorum evden çalışıyorum. [Emrah, 25][46]

As distinction between working time and leisure time is getting ambivalent over time, it can be claimed that, developers tend to see long working hours more acceptable. Their commitment to deadlines of project is increased because they don't want to be seemed responsible from any latency or problem of project delivery.

In this regard, autonomy and direct control should not be understood as opposite measures taken by management. Contrarily, these two are used in a subsidiary fashion. What we are trying to say is that developers may have autonomy over management of their working time and to take decisions related with technical tasks of production, while at the same time direct control at the form of deadlines is being executed upon them.

[46] Sometimes we have hard times in catching up. Sometimes we work in the weekends. It is fine if we don't but we do to relieve ourselves. There is no pressure on when to go to work. You don't have to be there at 12 sharp. I go there at around 1-1.30 but I leave at 9. But I keep working at home. Sometimes I don't go to the office but work at home. [Emrah, 25]

Where boundaries of technical and time autonomy given to developers is ambiguous, it becomes harder for them to know whether their managers or themselves are liable from the production process and potential flaws like not being able to complete assigned tasks in time or having technical problems more than admissible limits. They more likely to start questioning their abilities and competency, after complications arise one after another which they have difficulties to cope with. As tension gets increased with intensification of workload, concerns and vulnerability of them are amplified. When they have to put in extra hours to meet deadlines, this is their problem and it is experienced as personal failure because they did not meet the organization's norms (Rasmussen and Johansen, 2005, 92). Consequently, to prove their capabilities to themselves and to the others like co-workers and managers, they may willingly begin to work hard and long hours to meet the demands of management.

For example, at the beginning of his working life, Hakan told us that he had problems catching deadlines while responding demands of management. His manager assigned some tasks to him to be completed till deadlines without specifying technical details of work in a proper way. According to him, in addition to poor planning and management, responsible manager intentionally didn't help him to take advantage of his vulnerability.

> Bir de kendine yoruyorsun. Ben geç bitirdiğim için, benim eşekliğim ben öğrenmedim diye düşündüm. Hayal kırıklığı da oluyor. Ben mi beceriksizim yaptığım şey mi zor ayırtedemiyorsun, ki bence sektörde çok yoğun olarak istismar edilen bir durum. İlk işe girdiğimde dedim ya, adam bana işler verdi, çok öğretmedi diye, o ara tedirginlikle işi bitirecem diye baya mesai yapmıştım. Çok da belli etmeden yaptım yani, sanki ben kabahatliymişim de onu telafi ediyormuşum gibi. [Hakan, 26][47]

[47] You can't distinguish if it is your fault or the work itself is hard. I think this is constantly exploited in the sector. Since this job requires a mental work, once you cannot deal, you start blaming yourself of being incapable. You always face with these contradictions. As I told you, when I first started working, they assigned me some work but they didn't teach

As proposed by Rasmussen and Johansen, (2005, 92), the outcome of autonomy and direct control mechanisms is low professional self-confidence, but in this way workers tend to stay and try hard to meet the norms of management as long as they can deal with the harsh work intensity. Therefore, we can say that developers' freedom and autonomy is in reality the 'freedom' to meet the economic demands of the companies (Rasmussen and Johansen, 2005, 97).

Implementation of autonomous time management strategies requires compatibility related with the temporal and spatial aspects of production process. In addition to production organization, means of production should also be sufficient to make flexible working practices possible. Therefore, it should be required brief look over to the temporal and spatial settings of work in software development.

With new technologies and their enabling features of flexible working practices, boundaries of work seem to expand towards leisure time and space of software developers. Their lives are encircled by surrounding tools of software development, via computers, smart phones and etc. That means, developers don't need specific temporal or spatial settings in order to work. Thus, they become accessible and ready to work whenever or wherever they are needed. As told by Gökhan, in a blink of an eye, they can open their computers and start to work. Their life can easily be invaded at any moment by work with the help of these technologies:

> Sorumluluk büyüdükçe biraz hayatsal sınırlar, batı tarifi kapitalist sınırlar geçerliliğini kaybediyor. Hayatının içine biraz giriyor. Bir de açıveriyorsun,

---

me how to do. In that period, I worked overtime quite a lot because I was concerned with finishing the work. I didn't make it clear. I worked at home as well as at the office. The boss didn't know how much longer I was working after he left, as though I was the one at blame and the one to compensate. [Hakan, 26]

> çalışıveriyorsun. Bizim işin en önemli hikayesi o. Fiziksel bir atölye gerektirmediği için her yerde çalışabiliyorsun. [Gökhan, 44][48]

As indicated by Gökhan, distinction between work and leisure has been geting blurred with new technologies. Accessibility to production tools enable conditions which time autonomy strategies of management can be implemented more efficiently. Similarly, case of Ezgi who is working in a popular e-commerce website's system administration department is a good example of how flexibility in the sense of time and place setting of work augments the effectiveness of time autonomy strategies of managerial control.

> Hafif bir bilgisayar var şu an senle konuştuğum. Eğer uzun süreli bir yere gidiyorsam ve kalacağımı düşünüyorsam, 1 saat süresince erişilemeyeceksem, bunu da bildirmediysem hep yanımda oluyor bilgisayar. Şirket bana internet erişimi de sağladığı için herşeye bağlanabiliyorum.[Ezgi, 24][49]

It should be noted that case of Ezgi is a little bit different than others since her job includes consistent maintenance of website instead of pure software development from scratch. According to her job description, she is expected to take some precautions against potential problems which may interrupt functioning of website and to intervene unexpected problems immediately when they appeared. Variety of examples which she had to work from outside of workplace show us that to what extent her life and work are interlaced into each other.

---

[48] As the responsibility grows, these vital borders, Western-oriented capitalist boundaries lose their relevance. They intermingle with the life. You, on the other hand, you simply do, just start working. This is the most important story of our job. Since you don't need a tangible workplace, you can work everywhere. [Gökhan, 44]

[49] I have a light computer with me which I am using right now. If I think I will go out for some reason, if I will be out of reach for more than an hour, and if I haven't reported this, I always take this computer with me. Since the company provides internet subscribtion, I have unlimited access to everything which I need. [Ezgi, 24]

> İlişkileri etkileyebiliyor. Kendi istememde, öyle bir durumdan geçmiştim, orada allahın emri peygamberin kavli geçerken benim sunucuya, saçlarım topuzlu haldeyken bağlandığım oldu. Nişan ekranında erkek arkadaşım şey demişti, resimlerimiz, videolarımız falan geçiyor, iki ekran var, bir tanesine de nagios'u açalım, sorun olursa görürsün demişti. Bir ara da alarm sistemi vardı pagerduty diye kullandığımız, gece gündüz arıyor, önemli önemsiz. Gece 3 5 sürekli arıyor, rüyamda falan pagerduty arıyor. Rüyamda cevap verdiğim onun tekrar çaldığı oluyordu. Erkek arkadaşım, ben Ezgi ve çocuğumuz pagerduty yaşıyoruz diyordu. [Ezgi, 24][50]

On the other hand, flexibility and autonomy enabled by these technologies in regards of workplace and work time settings may be perceived by developers as rendering penetration of work into their life much bearable. When I asked Ezgi whether she has to be at the workplace whenever she is needed to solve unexpected problems, response of her was remarkable. She told us that she doesn't need to be thanks to the technologies enabling her to intervene problems at any place and any time:

> Yok, öyle olsa zaten çok çekilmez olurdu, evden müdahale ediyorum, yolda müdahale ediyorum, kafeden müdahale ediyorum. Fasıllarda, meyhanelerde, vapurda, insanlar koparken, elinde sürekli bilgisayar olan birisini görürseniz, o benim.[Ezgi, 24][51]

[50] I always have this in mind: what do I do now? It can impact the relationships. I have experienced during my boyfriend's family came to ask for permission. (kız isteme) While they were asking if I could marry their son, I was connected to the server with all the makeup and fancy hair. During the engagement ceremony, there were two screens: on the one, our pictures, on the other Nagios to see if any problem came up. For some time we used an alarm system, "pagerduty". It was ringing day and night, regardless of the reason. Its optimization was another process. I was dreaming pagerduty was calling. My boyfriend was saying "we are three at home: me, Ezgi and our child pagerduty". [Ezgi, 24]

[51] No, it would be unbearable. I respond everywhere, at home, on my way, in a café, in the bars. While people are having fun, if you see someone sitting in front of the computer and working, that is me. [Ezgi, 24]

It can be told that flexible working practices enabled by mobile technologies are common in Turkey. Taking an email or a phone call from managers related with the tasks have to be immediately done seems to be another routine which software developers have on their daily lives.

What should be stressed here is that, as flexible and autonomous working practices have implemented, working time of developers can be increased as well as the their productivity. Because, when developers have the freedom to arrange their working time, they tend to finish their tasks as much as possible, since quicker they finished, more leisure time they got.

3.3 Bureaucratic and Disciplinary Control Strategies of Management

Despite that all developers I have interviewed, complained about the work intensity, very few of them told that they expressed their discontent openly to their employers. When I asked Leyla why developers don't show resistance about overtime or intense workload against managers, she pointed out performance appraisals routinely made by management which are important for evaluation of wage raises and promotions.

> Bire bir de konuşulduğunda hiç kimse kanıksamış gözükmüyor. Ama hiç bir zaman ya arkadaş biz niye bu kadar mesaiye kalıyoruz, bu bizim suçumuz değil, ya da hani bunun parası gelecek mi, ayda beni şu kadar çalıştırabilirsin en fazla gibi bir itiraz kimseden yükselmiyor. Ama herkesi de anlıyorum, çünkü buna ses çıkardığın zaman ocaktaki maaş görüşmelerinde bile bu sana geri dönüyor, biraz daha kendini vererek çalışmanı bekliyoruz, o yüzden biraz düşük zam yapıyoruz diye konuşmalar geçiyor. O yüzden onlar da hakkını almak için aslında çalışmak görünmek zorundalar. Çalışıyor görünmenin yolu buna sesini çıkarmamaktan geçiyor. [Leyla, 25][52]

[52] On the other hand, I understand everyone. Once you raise these questions, you immediately see the reaction, even in the wage talks in January. They say "we expect you to work in a more motivated way. That's why you are getting a bit less wage increase".

Bargaining power of workers over the pay and other conditions in their individual contracts take its shape according to these assessments (Barrett, 2005, 80). Therefore, performance evaluations can be used as an instrument both for distribution of rewards and imposition of punishments in accordance with harmony and cooperation which workers show to their managers. Pay and promotions, after all, are more than just rewards for work well done. They are devices of control through not only bad behavior is punished, but proper behavior is rewarded (Edwards, 1979, 142). They structure recruitment, shape employee aspirations, and manipulate workplace behavior. That is to say, performance appraisal implies that doing what managers tell to do simply is in favor of individual workers since managers as superiors can influence their chances for advancement at working conditions (Kraft, 1977, 81). On the other hand, if workers show any kind of discomfort or disobedience, that the same appraisal can be used as a tool of punishment. Control over developers can be retained by this way which managers can maintain discipline of workers at their hand.

> Şu biraz sıkıntı, içeri girerkene kart okutuyoruz, çıkarken, öğle arasında hatta bulunduğumuz yerden çıkarken de, kattan inerken, kattan çıkarken, asansöre binerken. Naptın, nettin, nereye girdin, çıktın psikolojisiyle tuvalete girerken bile kart okutmaya çalıştığımız oluyor, alışkanlık ediniyoruz. Bu sana çok kontrol altında olduğun hissiyatını veriyor. Naptım, 9'da girmedim, 9:02'de girdim, bir şey olur mu ? Artı kapıdan girip çıkarken not alınıyor. Bu niye kötü biliyor musun, bizde performans primleri var aylık. O ay prim almadın, diyorsun ki ben bu ay 3 kere geç geldim, acaba bundan mı ? [Fırat, 30][53]

Therefore, employees need to seem working hard to get what they deserve. To seem hardworking, you have to stay silent. [Leyla, 25]

[53] This is also a problem: we have to swipe our cards while going in and out, in the lunch break, when leaving where you are, when going downstairs or upstairs, while getting on the elevator. After a while, psychologically, this becomes a habit; you even try to swipe while going into the washroom. This makes you feel you are under strict surveillance. "What have I done? I started shift not a 9 sharp but at 9.02. If anything happens?" Moreover, they take

In his study, Edwards defined this kind of control built into work rules, promotion procedures, discipline, wage scales and the like as bureaucratic control (Edwards, 1979, 131). To avoid direct, face-to-face, personal collisions between workers and employers which is almost inevitable with direct and simple control strategies, formal structures which exercised control through impersonal force presented as company policy or rules are established[R]. With this way, not only disputes between managers and workers are averted but also positive incentives are institutionalized under bureaucratic control, which workers identify themselves with, and committed to be loyal to the firm. Where goals and values of firm are internalized and legitimized by workers, force behind of the rules and regulations becomes invisible (Edwards, 1979, 150). Workers start to blame themselves when they fail to meet demands of management.

Furtherly, it can be drawn that, these rules are used for absorbing resistance and opposition of workers. Asserting all authority to these rules and regulations when they serve the demands of employers' interests makes it eaiser for managers to avoid workers' demands or complaints.

> Biraz bastırdık biz, maaş zammı yapılmadığı için, zam yapmıyorlar çünkü. Zamda da şeye bakıyor, Avrupa'da enflasyon yok sen niye istiyorsun ki diyor. Benim müdürüm İspanya'da mesela, adam anlamıyor Türkiye koşullarını. Türkiye'de böyle bir durum var, TEFE var TÜFE var diyorum, TEFE TÜFE o ne diyor. En son ara İK'yı da sor, Türkiye'de neler oluyor dedim. Ama anlayıp da anlamazlıktan geliyor. Bu tarz şirketlerin esprisi genelde bakış açısı bu. Hangisi karlıysa onu seçmek. O zaman 'Almanya'da 6'dan sonra mühendislerin çalışması yasak, bu bende neden geçerli değil'

---

notes when you get in. Do you know why this is bad? Our company has monthly performance premium. If you don't get your performance premium payment, you start thinking: "I have come late three times this month, maybe because of this?" [Fırat, 30]

> diyorsun, ‘e ama senin ülkende böyle bir şey yok ki’ diyor. İşine nasıl geliyorsa. [Semih, 44][54]

As indicated by Semih, when these rules and regulations don’t match up with interests of management, conflicts between workers and employers are generally resolved in favor of the latter. For these reasons, it can be proposed that regulations and rules in workplace, especially in global software enterprises are intentionally utilized against workers.

To conclude, it is obvious that the degree of profitability, surplus value production is determined in the workplace in accordance with the relations between employees and employers. In order to sustain surplus value production as well as to increase its rate, managers employ some strategies to take control of production process. Since software developers still have relative control over their labor, relations of control between workers and employees may demonstrate distinctive characteristics. Following that, managerial strategies of control may involve imposing direct control on workers as well as providing autonomy to them.

Till now, I mentioned that in what ways production organization of software development process has exhibited tendencies that strive to cheapen labor power

[54] We put some pressure on them for wage rise, otherwise, they would never care about the wage increase. For this raise, they say there is no inflation raise in Europe, why do you ask for raise? My manager, for example, lives in Spain. He doesn’t get conditions of Turkey. I try to tell him that there is such a situation in Turkey, we have consumer price index, wholesale price index, etc. He doesn’t know what I am talking about. Since I don’t know the economic terminology, I can’t express myself. In the end I asked him to ask PR to see how it works in Turkey. Of course, she does understand. But this is usually how this sort of companies work. This the common point of view: to choose whichever is more profitable. He sees everything according to his own interest. If he doesn’t see general inflation trend in Europe, he doesn’t give you any wage raise. But when you say, “well then, in Germany, engineers are not allowed to work after 6 p.m., why don’t you implement this in here?” He says “there is no such thing in your country.” Whichever serves his purpose better. [Semih, 44]

and to pass control over production process from producers to the hands of management. By considering that production relations and production organization are interdependent aspects of production process, I expect to see meaningful links between these two. In this regard, in following part of the study I am going to take a look on how software developers interpret their conditions which they are actively engaged in and take actions accordingly.

CHAPTER IV

SKILLS OF WORKERS AND JOB INSECURITY

So far, I have discussed means of production and technical organization of software development in Turkey, at first. Then, relations of production in the workplace have been explored in detail. In this chapter, workers' relation to their skills and qualifications will be discussed in two respects: firstly, job insecurity and future uncertainty heavily felt by the software developers will be outlined with its reasons and exemplified with reference to the statements of respondents. Secondly, strategies of software developers to discard or minimize this job insecurity and their relation to their own skills and self-development will be reflected.

The term "skill" is generally used in literature to refer the information and knowledge which workers need to be effective in their work, including the organization context and social relations that shape individuals' capacities to perform work (Benner, 2002: 27). In this definition, however, emphasis on economic and social aspects of skills seems to be lacking. What skill means for capital and labor is different from each other. For this reason, distinction between skill as task variety and complexity which is close to former definition and skill as task discretion in the sense of autonomy and control attained by workers has been generated within deskilling literature (Littler and Innes, 2003: 82). In my study, these two interrelated definitions of skill are going to be used according to the context.

Skills for companies in software industry are a lot more important than they are in any others. If a company does not supply necessary skills and knowledge required for production, it eventually falls behind in the innovation driven market. It may lost its competition power and even bankrupt unless adopts its organization to these conditions. However, rapid pace of transformation on production makes it harder to maintain and predict necessary technologies and corresponding skills which will be viable and valid in the future.

Due to this uncertainty, investing in workers and their skills constitutes several risks for organizations. Workers may simply leave company or invested skills and knowledge may become outdated. Such possibilities are obstacles on their way to rapidly respond shifting conditions of market. Short-term, project-based employment relations are more convenient for companies in this sense. As time term of employment decreases, uncertainty and risks to be taken by individual companies can be reduced and become more manageable. Like just-in-time, stockless model of production, there can be mentioned of prevalent tendency towards just-in-time employment relations (Bora et al., 2011, 24).

For workers, on the other hand, what looks impressive on their resume may easily become obsolete in a near future (Barrett, 2005, 174). The high speed of technological change and shifting market conditions in software development makes it difficult for workers to develop new skills and to stay on top of production technologies. Therefore, past achievements may easily be worn out and lost their value in market. As a result of constant extinction of skills, developers may face with fewer job opportunities and even unemployment possibility in the near future, if they cannot adjust their productive skills and knowledge to new market conditions.

Learning, as one of my respondents chose to verbalise his addiction to it as a "disease" is an everlasting process. There is no limit for learning activities, since there is no destination point to be fixated. In this sense, developers are challenged with a task of constant learning which is impossible to fulfill, neither in present nor future. Therefore, high level volatility of skills makes workers vulnerable in the presence of short-term, project based employment relations. Tension originated from this relation may reach very life-disrupting level if developer cannot renew skills and knowledge at his/her workplace/job. Therefore, what specified as flexibility for capitalists is experienced as tension by workers (Çerkezoğlu and Göztepe, 2010, 68), apparently as it is also case for software developers.

Management of skills, thereby self development is very common source of concern among developers; in a setting that job security corresponds to market validity of worker's skills and knowledge. Constant transformation of production organization which renders skills and knowledge required for production invalid in short span of time, puts job security of worker in a constant jeopardy. Consequently, this situation imposes severe pressure upon workers. They are consistently faced with risks of being redundant. Concerns about aging and future prospects in this regard are repeatedly articulated in interviews by interviewees. In addition, as mirroring of this tension, strategies that workers have adopted to secure their jobs are crucial to grasp dynamics of production process of software development and relations of production.

In our field study, it is observed that workers commonly embrace strategies which are showing market-oriented and individualistic characteristics. However, these market oriented strategies bolster the relations that lead to insecure employment relations, rather than to bring permanent solutions to the workers' problems.

In the light of these, the role of skills and its effects to the software production process are going to be delineated in this part. Firstly, I am going to examine reasons of uncertain future that exacerbate job insecurity fear among software developers. Subsequently, from what perspectives skills hold importance for employers and employees are going to be revealed along with findings from field study. Then, relation between management of skills and future prospects of workers will be evaluated.

4. 1 Job Insecurity and Uncertain Future

For many workers today, future is a source of hope and fear in the sense of working conditions. It is mostly uncertain and unpredictable. Software developers in Turkey are no exception in this regard despite their high qualifications and privileged position within general organization of production. That is to say, while on the one hand upward mobility is possible; on the other hand, these people may encounter with precariousness in terms of job insecurity and future uncertainty. In this part, I will give an account of the reasons of this uncertainty and job insecurity in software development industry in Turkey. In the following part, I will discuss the possible repercussions of this job insecurity upon the software developers as well as upon the relationship they have with their productive skills and qualifications.

Software industry in Turkey has a very short story. Most of the workers in this sector are younger than 30 years old. According to the survey by Kodalak (2007, 62), 69% of the software developers are not older than 30. In other surveys conducted by the Chamber of Computer Engineering in 2009[55] and Webrazzi in 2012[56] which is a popular technology blog in Turkey, the percentage of the workers under 30 are 63% and 88,2 % respectively. That is to say, only a small number of

[55] http://www.bimo.org.tr/wp-content/uploads/2011/08/AnketSonucRaporuv2.pdf

[56] http://webrazzi.com/2012/09/03/turkiye-it-sektoru-isgucu-raporu/

developers are performing this job at their middle ages. And majority of the older generation is employed in managerial positions rather than technical division of software development.

> Biz ilkiz bu işi bu yaşta yapan. Bizden öncekiler, dediğin gibi, ya çok az sayıdalardı altın devrin yaşandığı vakitlerde, yönetici oldular satışçı oldular. Bunları tercih etmeyip teknik devam etmek isteyen, kalan ilk insanlarız. Benim önümde hiç örnek, 50 yaşında bu işi yapan yok. [Semih, 44][57]

Given that, younger software developers are suspicious of their future since they do not know if they can still be employed at the technical division in their later ages. Today, no software developer can clearly see what will happen at their later ages if they remain as technical workers in this industry. In the following parts, I will explore the reasons which lead to this job security concerns.

4.1.1 Ageing and Skills

One of the reasons of uncertain future in the sense of employment seems to be emanated from the opinion that aging has an inevitable detrimental effects on technical capabilities of software developers. As the majority of my respondents agrees on, skills and knowledge, hence performance related to technical division of software development is decreasing as inversely proportional to age. Regarding division of labor, Gökhan draws evaluation about aging and its future effects. Being part of a kitchen, which means directly handling with codes and doing programming is really hard after some age. After a point, mind gets tired and starts losing its full-functioning. He reckons for this reason, developers have to be prepared for their future and move to managerial positions until a certain age.

[57] We are the first generation who do this work at this age. Previous generation, they were really a few, it was the golden age. They have become managers, salesman. We are the first ones who don't prefer those, who prefer to continue in the technical part. I have no one to follow. There is no one who can still do this at the age of 50. [Semih, 44]

> Bu da sorularında vardır muhakkak. Ben nerde yazmayı bırakacağım ? Fikir adamı olarak devam edeceğim bir noktası var bunun. Futbolcu gibi değil, balet balerin gibi değil ama bunun da bir yaşı var. Bence 50'den sonra falan çok dikkatin dağılır diye düşünüyorum. Şu an bile fazla da ... İnsanları örgütlemede daha verimli olabilirim. Tasarla, dağıt, yönet hikayeleri. Yazılım sektöründeki insanların da kendilerine böyle bir yön çizmeleri gerekir. Eskiden aklıma gelirdi de özellikle üstüne düşmezdim. İnsanların belli bir plan çizmeleri lazım. Yükselme planı olabilir, kariyer planı olabilir. Tabi ki bunu 20'li yaşlarında bilmeleri mümkün değil. [Gökhan, 44][58]

The view that the technical competency and working performance is started to diminish after certain age seems to be very common among developers. Emrah thinks the same about this situation.

> Emeklilik yaşı 65 ama ben 45 yaşından sonra kendimin yazılımcı olarak çalışabileceğini düşünemiyorum. Dikkatin dağılacak, eskiyeceksin, entropi var yani evrende. [Emrah, 25][59]

Supporting this very same point, Cihan says that software workers lose their capabilities starting from the age of 35.

> Yazılımcılar 35 yaşından sonra akıllarını yitirir, eski performanslarıyla çalışamamaya başlarlar. Bir kaç makalede okuduğum bir şey bu. Yazılımcıların performanslarının yaşla düştüğü söyleniyor. Dolayısıyla belli

[58] Probably you will ask this as a question: When will I stop coding? There comes a moment that I will move on as an intellectual in the field. Maybe not like a footballer or like a ballerina but this also has an age limit. I think after 50 years old, you get highly distracted. Even now it is too much… I can be more efficient in organizing the personnel. Designing, allocating, managing sort of jobs. In software sector, people need to make your own way. I used to think about this but I wouldn't care much. People need to have plans; this might be a plan for promotion, for their career. Of course, they can't know this when they are 20. [Gökhan, 44]

[59] Retirement age is 65 but I can't imagine myself as a software developer at 45. You will get distracted, you will get older, universe has the law of entropy. [Emrah, 25]

> yaştan sonra yazılımcıyı işe almayı çoğu insan istemez, en azından developer olarak. Olacak da idari pozisyon seni işe alacaklar. [Cihan, 29][60]

However, it is hard to say that age is the real factor which affects negatively their performance and leads them to be less capable or "useless" at the end. Hence, some other explanations can be put forward about relation between aging and technical competency. As Sennett points out, this view might be a socially constructed prejudice for justifying and obscuring destructive effects of labor market economics (Sennett, 2006, 21).

In software industry, wages of software developers generally increase as the developers have seniority. But that also means, senior workers might become expensive resources for companies over time. When we look at the previous studies, because, it is observed that cost of employing and training junior developers is cheaper than retraining older developers in this industry, if needed skills are not special or qualified that much (Kodalak, 2007; İliç, 2011). Therefore, for employers, employing younger developers might be more viable option in accordance with labor market economics. Statement of Yüksel proves that this kind of approach is common among employers in Turkey.

> Şey her zaman var tabi. Evet, adam diyor ki, senin çok güzel yeteneklerin var ama sana bu kadar para vereceğime 2 tane junior alırım, başına da senden daha az maaşla çalışan, elimde hali hazırda bulunan görece tecrübeli birisini koyarım, hem işi öğretirim, hem de daha fazla iş yaptırırım diyor. Haklı olduğu yerler de var, atladığı yerler de var. Olaya tamamen rakamla yaklaşmaya başlandığında, bence kaybetmeye de başlanıyor.[Yüksel, 40][61]

[60] Software developers lose their minds after 35, they can't work with the same performance as they are young. This is something I have read in couple of articles. It is commonly said that software developers they lose their working capacity. Many employers don't want to hire an older developer, at least as a developer. They may hire you for administrative positions. [Cihan, 29]

[61] The employer says "well, you are highly talented and qualified. However, with the money I will pay you, I can employ two junior workers. As the manager, I can find an

Later on, I asked Yüksel, whether he could easily find a job right now, if he resigned from his job or he is laid off for some reason. He told us that it would be a painful process for him due to the reasons above,

> Tecrübenize göre değişiyor. Mesela, ben 15 yıldır çalışıyorum, benim için çok sıkıntılı bir süreç. Bir taraftan 15 yıldır aynı yerde çalışıyor olmanın getirdiği rehaveti ve rahatlığı kırmak anlamında, bir de high qualified bir noktadasınız, geçiş yapabileceğiniz işler de çok fazla değil. Tecrübenizi kullanabileceğiniz, faydalı olabileceğiniz, sizi de tatmin edecek bir şeyler arıyorsunuz. [Yüksel, 40][62]

Despite her rich working experience, Sevil proposed that she have already passed through such distressed period while looking for a job when she was 35. For her, it would be much harder, if she had to look a job now, when she is even older.

> 35 yaş civarına gelmiştim, o yaşta iş bulmak da o kadar kolay değil. Hele şimdi çok daha zor. Büyük şirketlerin know-how'ı var çoğunlukla, kendi adamını kendisi yetiştirebiliyor. Yöneticilerini de dışarıdan transfer edebiliyor ya da kendi içinden yetiştiriyor. Seni dışarıdan alması gibi bir şey yok. [Sevil, 40][63]

---

experienced employee who is already working for me. I can both train them and make more profit by spending less money." She is right to some extent yet he overlooks at some points: if you develop a money-oriented approach, you start losing, [Yüksel, 40]

[62] It depends on your experience. For example, I have been working for 15 years now. This is a very troublesome process for me. On the one hand I have to get rid of the lethargy and sluggishness of having been employed in the same place for 15 years. On the other hand, you are a high-qualified worker, it is not easy to find a job to transfer. You are looking for some works that you can utilize your experience, that you can be satisfied. [Yüksel, 40]

[63] I was 35 back then, it is not easy to find a job at that age. Now it is even harder. Big companies usually have their own know-how section. They can train their own personnel. They can transfer for managerial positions or train them as well. They don't need to employ you as the outsider. [Sevil, 40]

Therefore, what is worn out with the age may be the market validity and viability of older developers' skills rather than their real technical abilities. For Yüksel, it is the "commercial logic" that leads companies to the easily withdraw their substantially experienced and skilled workers, that is, senior and older workers. For example, Semih who is working in a leading international software enterprise told us that if he had been younger with his present skills and knowledge, he could have handpicked a job with good salary among several options. Though, he doesn't think that he can do the same today because of his age.

> Ya genç olsam kesin var zaten. Şu an 30 yaşında olsam süperdi, istediğim yere istediğim fiyata gidebiliyor olurdum. Tabi yaş dedim ya, onu ben de kestiremiyorum. Kim ne kadar beğenir, görür bizi, bilmiyorum. [Semih, 44][64]

Therefore, the reason of this job insecurity and concerns about the future do not only emanate from the common belief that ageing is detrimental to technical skills of the developers. It also is constructed by the market conditions that destroy the validity of the skills of the older software developers. Market conditions are determined mainly by two motives: market compels the firms to hire developers who can more easily adapt to the changing market conditions firstly, and then less costly ones, thereby, the young developers.

4.1.2 Vulnerable Market Conditions of Software Companies

Vulnerable market condition of small and medium level companies is another reason which exacerbates the concern about future for software developers who are employed in such workplaces. Great numbers of medium and small scale companies

[64] If I were young, that wouldn't be a problem. If I were 30 now, I could find any job for the wage I ask for. But, as I said age, I can't make out now. I don't know how much longer I will be appreciated. I have some colleagues who are 4-5 years older than I am. They are still very well-known in Istanbul, Ankara. They are still working. [Semih, 44]

in Turkey barely afford expenses and survive with the help of state funds. Unable to apply long-term and more sustainable strategies, they are prone to head towards short-term outputs. As sensible to the volatile market conditions, companies on these scale are on thin ice, their future seems to be full of dangers and uncertainties, as well. For that reason, job turnover rates in these companies are quite high. Generally, in their first years of working career, junior developers prefer to work in those in order to gain experience. But for older developers who need long term job security and relatively peaceful, trouble-free working environment, working in these companies is not the best option. As Isıl also points out,

> Küçük şirketler inanılmaz kötü. Bilişim alanında teşvik dağıtılıyor çok fazla. Her önüne gelen şirket açıyor. İçinde 3-5 kişi var. Ben hatta tweet atmıştım, kapitalizm kapitalizm olalı böyle bir kalitesizlik görmedi diye. Kesinlikle hiç bir süreç falan uygulanmıyor, her şey parayı, teşviği alana kadar. Test bile yapılmıyor. Müşteriye inanılmaz kalitesiz bir ürün veriliyr. Sürekli bir kazık atma hali. Küçük şirketler bu durumda anlayacağın, bugün varlar ama uzun vadede ne olacakları belli değil. 3 şirket değiştirdim bu şekilde. Hepsinin de sahipleri akademisyen kişilerdi, hepsinde de bunlar vardı. [Sevil, 40][65]

Therefore, developers who are working for such firms feel obliged to monitor whether company can achieve new projects or not. Stability in this sense is important for them in order not to find themselves as unemployed from one day to the other. Working for more institutionalized companies, for this reason, is thought to be more secure in regard to future employment continuity. Both Metin who is working in the medium level company and Murat who is working in the big-scale and institutionalized company have similar thoughts supporting this claim.

[65] I can't find job in big companies, smaller businesses are terrible. Too much incentives is given in software sector. Everyone launches a business. 3-5 people work in there. I tweeted once: "Capitalism has never encountered with such an unqualified business." There is no procedure, never. Everything looks good on the paper, until you get the incentive (funding)! Firms are not even tested. Product is of poor quality. Everyone tries to rip off the other one. This is how small businesses are. I have changed three jobs. All of them were owned by the academicians. [Sevil, 40]

> Şirket yeni işler alıyor mu gözlemek durumunda oluyoruz. Atıyorum, Siemens çalışanı olsan sanmıyorum gözleyeceğini, batacak değiller sonuçta. İşte küçük firmaların da handikaplarından birisi bu. [Metin, 33][66]

> Kurumsal yere girme sebeplerimden birisi de bu. Burada 10 sene daha çalışırım. Burası da bu istihdamı sağlayabilir bana. Ücretim artsa bile bunu karşılayabilecek gücü var. Burası güvenli bir liman benim için. [Murat, 28][67]

As the statements above illustrate, working for small and medium level companies contribute to the job insecurity. These companies cannot guarantee a continuous and well-established job for their employers. On the other hand, majority of the software development market is composed of small and medium scale companies[R]. Thus, software developers are forced to work for these companies despite the lack of job security, especially for long-term employment. Since these companies themselves are insecure in the market, they cannot provide or guarantee any kind of long term employment security for their employees.

Keeping in mind these declarations, software developers put the fact that bigger companies provide job security for their employees more than the small and medium companies. As far as my field site illustrates, software developers strongly believe that bigger companies are more secure in the market. They are not as fragile as the small or medium scale companies. However, with the advent of new software service models that are increasingly used by the bigger companies, job insecurity is reproduced from another perspective. This incoming software service model leads the way towards the transformation of the software development from a productive work model to a service model. Software developers, who are employed in bigger

[66] We have to consistently check whether the company could take new projects. If I were a, let's say, Siemens employer, I don't think I would care about this. They are not going to bankrupt anyway. This is the handicap of small companies. [Metin, 33]

[67] This is one of the reasons I started working in a corporate firm. I can work 10 more years here. And this company is capable of providing employment for me. It can meet my demands for raise. This is a safe haven for me. Murat, 28

scale companies, turn out to be software service users rather than direct producers of the software which discloses the indications of a new revenue model.This transformation observed in this new model of software is highly occasioned through job insecurity led by service models.

4.1.3 Effects of Software Service Models

Service models are getting popular and dominating global ICT market categorized as follows; software as a service, platform as a service and infrastructure as a service (Dubey and Wagle, 2007). With these models, buyers/users like non-ICT companies become more flexible to switch vendors and have few troubles in maintaining software (Turner et al., 2003) while increasing their control over production and labor. Therefore, most of the leading enterprises tend to incorporate these service models. As the quotation below illustrates, software becomes a subscription requiring service which is constantly updated and maintained by service providers.

> Bir de artık piyasa şartları değişti, şu an yazılım almıyorsun, hizmet alıyorsun. PAS diye geçer, Platform As a Service. Baya sana platform satıyor, aylık ücret alıyor. Sen eskiden napıyordun, bir tane server alıyordun, ona internet bağlatıyordun, kullanıyordun. Bir kere para veriyordun, şimdi aylık veriyorsun. Gelir modelleri çok değişti. Genelde hizmet alıyorsun, kafan rahat ediyor. Yok sistem yanmış yok şu olmuş dert etmiyorsun. [Erdem, 23][68]

[68] Besides, market conditions have been changing now. You don't buy the software any more. You buy service. It is called the revenue model of Heroku, PAS, Platform as a Service. They simply sell you a platform and charge you on the monthly basis. Before this, you used to buy a server, connect internet and use it. You used to pay once. Now, you need to pay monthly. Revenue models have changed a lot. You usually buy service, you don't need to think about if the system has a problem. [Erdem, 23]

By means of this, companies are not in the need of as many software developers as they used to with the aim of producing and maintaining software applications. Hence, such models potentially enable software user companies to outsource some related tasks in order to reduce ICT costs. Companies for this reason favor this model rather than developing software by themselves. Available jobs in heavy information and communication system user organisations like banks or public services seem to be diminished as a result of growth in outsourcing in near future (Magdoff and Magdoff, 2004, 28; Flores and Gray, 2000, 15). Therefore, job opportunities for developers are going to be rooted into two options; either switching to specialized software enterprises or staying non-ICT organizations and working as software service users. Erdem conveys his observation regarding this division:

> Onlar ne yapıyorlar biliyor musun; Hakan vardı şurada çalışıyor normalde. Onlar ya o tarz [hizmet sağlayıcı şirketlerde] yerlerde işe giriyorlar, ya da bunları kullanan insanlar oluyorlar. [Erdem, 23][69]

As software developers are employed as software service users, the quality of work tasks' content, which are under the responsibility of software developers impoverishes. Later on in our conservation, I asked whether utilization of these services require specialized expertise, Erdem answers:

> Şey olarak düşün, eski kafayla gidersen, burada 5 tane proje varsa, 5 tane sistemciye ihtiyaç duyuyorduk. Şimdi bir tane devops hepsine fazlasıyla yetiyor. O adamlar biraz kabuk değiştirdi. Artık direk metalle uğraşmıyor yani makinayla uğraşmıyor, bu hizmet servisleriyle uğraşıyorlar. Onları yönetmek haklarında bilgi sahibi olmak da baya iş. Alan hızlı değişiyor. Değişime uyum sağlamak önemli. [Erdem, 23][70]

[69] Do you know what they do? Hakan, for example, normally he works there. They, either find a job in those kind of places or become users of these services. [Erdem, 23]

[70] Back in the days, when we had 5 projects here, we would have 5 system developers. Now, one developer is more than enough for all. Those guys have changed shell. They

Erdem's statement underlines the tendency of how job descriptions as well as skills required for software developers is impoverished. Since self-development in the fast-changing market conditions is very important, software developers who work as software service users are in a more vulnerable situation. They hardly find the opportunity to develop new skills. Thus, those developers feel more insecure than their other colleagues. Moreover, with proliferation of software services, it becomes possible for service providers to outsource these jobs to other countries where cost of labour power is cheaper. Therefore, in the near future, developers may face consequences of this change, as Semih mentions:

> Geçen bir müşterime gittim, yazılımcı işte. Abi cloud'u okudum ben dedi, ee dedim, biz işsiz mi kalacaz abi dedi, siz mi yapacaksınız bizim işleri de ? Oğlum dedim, o cloud işi olursa ben de işsiz kalacam, sen de kalacaksın. Çünkü niye beni tutsun adam, çok daha ucuz hintlisi var çinlisi var. Çinli olmaz belki de, Romanyalısı var. Niye bizi tutsun. Bir 10 sene daha kurtarır bizi, o vakte kadar gelmez. Ama geleceği kesin bu ya da benzer bir teknolojinin. [Semih, 44][71]

"Lucky" developers to be employed by service providers, on the other hand, might face with more intense work loads to keep their job. Flexibility acquired by managers and companies result in "work tasks that are designed as that far fewer workers are needed to get same amount of work done" (Greenbaum, 1998). Semih's experience is exemplary in this sense. After his company made a move to expand its

---

don't deal with the metal or with the machine directly, they care about the services. However, it is a big deal to be well-informed about their management. Field changes really fast. It is important to keep up with the time. [Erdem, 23]

[71] The other day, I went to see a customer, he is a software developer. He said, "I read about the cloud. Will we be unemployed? Are you going to overtake what we do?" I told him "If that cloud thingy works, I will also be unemployed as you will. Why would the employer hire me? He can hire a Chinese, Indian or Romanian worker; they are much cheaper." Maybe 10 more years we can work until Cloud arrives. But it is definitely coming, this or a similar technology. [Semih, 44]

market share by integrating different products under one package, responsibility and workload of Semih has increased implicitly, while subsidiary companies and developers working over there has been knocked down out of market.

> Daha önce kaç kişi ekmek yiyordu buradan. Şimdi 2 3 kişiye kaldı ekmek. Üstüne üstelik benim de iş yüküm artıyor. Ben farketmiyorum yükümün arttığını, çünkü neden, bana artık buna da bakacaksın diyorlar. Ama zaten 8 saat çalıştırıyor. O anlamda daha yapabileceği bir şey yok. Yeni sistemleri öğreniyorum işte. Dediğim gibi diğer firmaların da mühendisi varken, 3 kişinin yerine tek kişi bakıyor artık. 3 uzmanlığı benim üzerimde toplayıp oradan kar ediyor. [Semih, 44][72]

4.1.4 Upward Mobility: A Hope to Get Managerial Positions

In parallel with the issue of aging and changing market conditions, staying in technical division of software development seems to provide very little job security regarding future. Recall that, for many software developers, staying in the technical division of production after certain age is hardly possible for various reasons mentioned above. Along with ageing, the significant change in market conditions brings about job insecurity by deploying new market models which need less employees. In this respect, software developers strongly believe that being promoted to the managerial positions is a way of ensuring long term job security. This, as a strategy to remain competitive in the labor market, is thought to be most secure way, especially by the younger developers. For Emrah, his future in this regard is very clear; either a position in management or unemployment.

[72] Couple of years ago, many people would make their livings out of this job. Now, only 2-3 people get employed. Moreover, my burden increases. I don't realize that my workload increases because they ask me to tackle with one more thing. But they employ you 8 hours in a day. In this respect, there is nothing more they can offer. I learn about the new systems. As I said, it is the same in other companies. Now, one person does what 3 employers used to do. Company makes profit by putting the workload and expertise of three people on me. [Semih, 44]

> Eğer, yönetici olarak çalışmaya devam etmediysen, ya da kendi işini kurmadıysan işsiz kalacaksın. Bu yani. Opsiyonlar bunlar ya kendin iş kuracaksın ya da yönetici olarak devam edebileceksin. [Emrah, 25][73]

In software development organization, managers basically lay out time scheduling and take fundamental decisions about technical design of projects. In addition, they participate to the allocation of workers' task and supervision of whole process. Technical workers, on the other hand, are responsible for executing these tasks assigned by managers. What happens in general, as Hakan tells us, people start working in latter group mostly as junior developers, achieve promotions and climb up the career ladders throughout the years. However, ascending to managerial positions may not be attainable for each of them.

> Genel olarak nasıl oluyor biliyor musun ? Adam 4-5 sene çalışıyor, başka yere yönetici olarak geçiyor. Ya da çalıştığı yer büyüyünce, orada bir yönetici gidiyor, artık onu yönetici yapıyorlar. Çok da belli olmuyor bunlar. Baktığında önünü göremiyorsun. Çok da bakmıyorsun şu an için. Ben bakmıyorum yani. ama 3 sene sonra problem olabilir. [Hakan, 26][74]

In a sector, within which growth is not necessarily proportional with rise in employment, available managerial positions are hardly predictable. In the course of organizational change, it seems that horizontal as well as vertical division of labour allocated formerly managerial tasks to a wide range of agents starting from workers to the customers. That is to say, proliferation of tendencies within managerial policies to celebrate communication and collaboration between workers, managers, departments and customers make it possible to access information without

[73] If you don't continue working as a manager or if you don't launch your own business, you will be unemployed. This is what it is. These are the options you have: either become a manager or set up your own business. [Emrah, 25]

[74] Do you know how it usually works? Someone works in a company for 4-5 years. She transfers to somewhere else as for manager position, or as the company grows, she becomes manager there. But these are not certain things. You can't see the future. I don't care this. But this will be a problem in 3 years. [Hakan, 26]

functioning of mid-level managers (Thompson and Warhust,1998). Control and cooperation mechanism of bureaucracy seems to be replaced by control and co-ordination of diverse forms of labour which are divided more horizontally (Greenbaum, 1998:138). Hence, the need for mid and low level managerial positions gradually disappears. Obviously, this transformation results in further concerns in younger developers. Because, younger developers are forced to encounter with a dichotomous situation: either staying in technical division which is not a welcoming place for older workers or seeking for managerial position which is hard to get in. Therefore, neither of the options provides job security. Murat's projection concerning the difficulties of being promoted to managerial positions seems to show such apprehension about the disappearance of managerial positions thereby about the future uncertainty.

> Valla bu biraz bizim dönemimizde belli olacak. Bizim nesil belirleyecek. Sektör görece yeni bir sektör. Şu an 20 30 senedir piyasada olanların hepsi yönetici konumunda şimdi. Bu işi yapan sayısı azmış zamanında. Fakat bizim 20 30 sene sonra ne olacağımız belli değil. Hepimiz yönetici olamayacağız sonuçta. O kadar bir şey, pozisyon yok, olması da mümkün gözükmüyor. Ama firmalar için belki daha pahalı kaynaklar olucaksın. İşlerine yarayacak adam olacak mısın belli değil. Nasıl diyim, bunu biraz bizim nesil gösterecek. Çok öngöremiyorum bu durumu. [Murat, 28][75]

Almost all of my interviewees think that moving into management positions provides a resolution against job insecurity. However, they also mention how obscure it is to hold a managerial position in the future. They cannot even presume (estimate) the possible need for these positions. Previously, software developers

[75] Well, this situation will become clear with our generation. Our generation will determine it. This sector is relatively new. Today, all of those who have been in this sector for 20-30 years are managers. Back in the days, number of employees in this sector is low. But we don't know what to do in 20-30 years. Obviously, we can't all be managers. There is not that many positions, it is not very likely to be by that time. But maybe you can be an expensive resource for the company. It is not certain if you will be, I cannot foresee that. [Murat, 28]

used to have more chance to move into managerial positions. However, the number of software developers are increasing deuce of a lot and hence, it seems less likely to hold managerial positions for these developers. To conclude, software developers in Turkey endure future uncertainty in regards to job security as a result of ageing and changing market conditions. This situation seems to emboldens their tendency towards self-reliant, self-developmental, hence self-exploitative strategies for job security.

With threat of redundancy hence potential unemployment, not only present but also future dispersed into present are started to be appropriated by capital (Ehrenstein, 2006 cited in Papadoupoulos et al., 2008). With this way, appropriation intensity of surplus value can be increased as developers incline to self-exploitative actions for both their present and future employment contract, as Papadoupoulos, Stephenson and Tsianos (2008, 233) explains :

> Self-exploitation happens in the regime of precarious life and labour when someone tries to anticipate and explore the future through its dissemination into the present and to intensify their own efforts to ensure that they remain competitive in the future. This post-contractual form of dependency is twofold: it is a dependency on the employer, who offers limited contracts, as well as a dependency on oneself to increase one's own capacity to get such contracts in the future. (Papadoupoulos et al. 2008, 233)

Software developers, especially those who work in technical division, put more and more effort to catch up with the newly developed skills and technologies which are determined by the market conditions. In following part, I will explore how these strategies related to the skills gain a seat in software industry.

4.2 Skills and Skills Management Strategies Adopted by Workers

Future of software developers in the sense of job security is uncertain in this industry. Consequently, tendency to rely on individual skills is very common among

developers. Such that, these skills are determinative on market value of labor power. Skill renewment, in this sense seems to become the only assurance of job security. Under these circumstances, therefore, to remain productively employed, software developers consistently manage and renew their skills in regard to market conditions. Besides, they stay in touch with many other individuals and firms in the broader industry (Benner, 2002, 48). Skills renewal pressure upon software developers exacerbates if they cannot foresee their future in the sense of job security. Accordingly, the degree of workers' self-exploitative actions intensifies. This situation can be observed through relations of workers with skills and strategies of skills management they have adopted.

In order to understand the process of skills management, we need to explore through which market parameters validity of skills as well as strategies of developers are determined. This is of importance in the sense that strategies chosen by the software developers are also shaped in accordance with these parameters. In this section, I will firstly, explain what "skills" mean in software industry and how its market validity is determined. Secondly, I will explore how these strategies and parameters are constituted and employed by the developers in addition to the effects on relation of developers to their skills and working conditions.

As I briefly discussed in the chapter I, production instruments are crucial part of software development. Beginning from the scratch, developers need to apply various kinds of programming languages, paradigms and Computer Aided Software Engineering (CASE) tools in development process. Qualifications and competencies on these instruments, therefore, cover great majority of software development requirements.

While deciding which production technologies will be implemented in a software project, companies generally pay attention to utilize widespread technologies.

Because, for companies, rare technology means rare labor power available in the market, in addition, less information and documentation about these technologies. Murat confirms this reflection:

> Az çok herkes aynı şeyleri kullanıyor. Ya Java kullanırsın, ya C# kullanırsın, ya Oracle kullanırsın ya MsSql kullanırsın. Genel olarak öyle çok aykırı teknolojilere giden yok. Bunun da pek çok sebebi var. Bir, aykırı teknolojiler demek işi bilen daha az kişi demek, daha az dökümantasyon demek, daha az destek alman demek. İki, ben bunu öğrenecem ama bu benim dışarıda hiç bir işime yaramayacak diye düşünüldüğü için, onu öğrenmek isteyen, çalışacak adamı da zor bulursun. [Murat, 28][76]

If companies utilize rare and old technologies, they may face with difficulties while supplying labour power. Excessive dependency on labor, however, may thwart their control over production process. However, capitalists want to keep labor as disposable and/or easily replaceable part of the production process. For this reason, they prefer utilizing widespread production instruments and technologies.

> Yeni ürünlerde, kişilerden bağımsız olması açısından atıyorum biz bunu yeni teknoloji bir şeyler, nova ile yapalım, Ahmet'e Mehmet'e bağımlı kalınmasın, jenerik bir şey olur, bakımını daha kolay idare edebilirsin diye düşünülüyor. O yüzden 0'dan başlayan projelerde buna dikkat ediliyor. [Kazım, 41][77]

[76] Everyone uses more or less the same things. More or less. You either use Java or csharp or oracle or mssql. People rarely prefer extreme technologies for many reasons: Firstly, extreme technologies means less people know about it, it means they are less documented. Therefore, it means you can get less support and incentive from that company. From employer's perspective, it is less useful. It is harder to find an employer who wants to learn that. [Murat, 28]

[77] In the new products, independence from individual workers is also a concern of managers; they think "let's do this with nova so that we won't be dependent on specific people, we can improve and manage it more easily if it is something generic." That's why people pay attention to this while kicking off a new project. [Kazım, 41]

By using widespread technologies, therefore, companies can exploit and hand-pick workers from a larger pool of labor. Thereby, they obtain flexibility through which they can adjust to the changing market conditions.

Not unlike to the companies, for workers, keeping up with the newly developing and prevalent technologies is a measure to render their skills compatible with the labor market. For the developers who need to ensure their future employment, prevalence of current technologies promises the future use of same technologies, thereby, guarantees the demand for labor with corresponding qualifications. Supporting this idea, majority of my interviewees state that having skills on widespread technologies is more reliable for their job security. That is why almost no one in software industry want to and prefer taking place on firm-specific projects in which rare and outdated technologies are used.

> Sadece S'nin geliştirdiği yazılım teknolojileri de var. Onun üzerinde 20 senedir devam eden bir proje olabiliyor. O projede çalışan kişi sadece o teknolojileri kullanıyor. Başka bir açılım noktası da yok. sektöre çıktığı zaman sektöre, şunu biliyorum demesinin kendisine çok fazla faydası, getirisi de olmayabiliyor. Eğer bu dediğimiz projede çalışan arkadaş, kendini farklı alanlarda, ama kişisel çabasıyla geliştirmediyse, yarın bir gün iş aramaya kalktığında iş tecrübesi anlamında, ben c++'da, java'da, .net'de proje yaptım diyemeyecek. O yüzden genelde bu tip projelerde çalışanlar, başka projelerde de paralel çalışıyım, orada da bir şey öğreniyim, ya ben burada kendimi çok geliştiremiyorum, yarın bir gün bu proje gitse ben ne yapacağım gibi endişeler içine düşebiliyor. [Kazım, 41][78]

[78] There are some software technologies that are developed only by S. Some projects have been working on these sort of technologies for more than 20 years. The employer who works in this project uses only these technologies. If this employer working in this project hasn't invested in his self-improvement, she will lack in terms of work experience; she won't be able to say "I have conducted projects on C++, Java, .net". Thus, those who work in these kinds of projects are more likely to be concerned about improving themselves. [Kazım, 41]

As indicated by Kazım, having skills on rare and old technologies is a disadvantage for workers in the sense of remaining competitive in labor market. Therefore, pressure of job insecurity is more heavily experienced by these developers. In this respect, vulnerability of them reveals itself as a concern of self development and renewment of skills. In order to remain competitive and employable in labor market, they are more prone to force themselves to keep their skills up to date.

> Öyle bir endişe oluyor, eski teknoloji kullanan arkadaşlarda. Genelde herkes, gördüğüm kadarıyla, en güncel en yeni teknolojilerin kullanıldığı projelerde çalışmak istiyor. Yeni, en yeni teknoloji neyse, yeni bir proje başlıyorsa ben de çalışabilir miyim o projede, haklı olarak düşünüyor, gündemden geri kalmamak için. [Kazım, 41][79]

Therefore, market prevalence of technologies employed in software projects is perceived as most reliable criteria on which particular skills can provide long term job security to its possessors and which cannot. Even though working on rare technologies may provide short term advantages, concerns for long term job security seems to outweigh the immediate advantages.

Validity of qualifications and skills that software developers have, also vary in accordance with age and wage of workers. Because, in the case that there is no shortage in labor market in the sense of skills needed for production, companies can employ workers according to their age and wage instead of their technical competencies. Thus, we need to know how software projects implemented in Turkey shape the need for high qualifications and skills in labor market.

[79] Employees who are working with such old technologies have such concern of not renewing their skills. Therefore, as far as I see, people prefer to work in projects in which latest technology are deployed. When a new project initiates, everyone rightfully wants to work in those. This concern comes up for not lagging behind the latest trends. [Kazım, 40]

Majority of software projects in Turkey are implemented for general purposes which rarely require specific, profound skills and expertise. Therefore, companies can execute production and generate software output with workers whose majority has mediocre and average skills.

> Türkiye'de çok niş işler, kaliteli işler çıkarılmaz. Bu yüzden ihtiyaç da olmadığı için bir yerden sonra hep rutin işler çıkarmaya başlarsın. Yurtdışında sürekli ileri gidiyor adamlar. 5 yıllık ile 8 yıllık yazılımcı arasında o yüzden fark var orada. Ama şahsi fikrim Türkiye'de yok. [Cemil, 34][80]

> Sektörde aynı konu üzerinde çalışan 5 senelik mühendisle 15 senelik mühendis arasında çok bir fark olduğunu ya da olacağını düşünmüyorum. O kadar zor değil hiç bir şey. [Murat, 28][81]

This situation indicates that skills of workers in this industry regarding seniority and working experience do not necessarily produce market value for its owners. Hence, companies can employ workers according to their costs rather than how much skilled and experienced they are.

> Birisi bana iş başvurusunda bulunsa, biri 5 yıllık birisi 8 yıllık, tecrübesine değil başka faktörlere bakarım. Türkiye'de böyle değil ama. Böyle iki kişi gelse ilk olarak maaşına bakarlar, çok para isteyip istemediğine. [Cemil, 34][82]

[80] In Turkey, software projects which are execured are rarely niche; it is hard to stumble upon a qualified work. That's why after some time, you start doing routine works. People abroad, on the other hand, they always advance. That's why 5-year and 8-year experience are different. To my mind, in Turkey there is no such difference. [Cemil, 34]

[81] I don't think there is a big difference between a 5-year experienced engineer and 15-year experienced engineer. Nothing is that difficult. [Murat, 28]

[82] If two people apply me for job, one is 5 year experienced, the other is 8 year experienced, I don't care about the experience. I check other factors. But this is not how it works in Turkey. They first look at the wage, at how much money they ask. [Cemil, 34]

Technical organization of the software development at companies is also significant in terms of the relation of developers to skills and qualifications they have. Because, forms and features of assigned tasks differ from one production organization to another. To illustrate, in bigger scale companies where division of labor is more strict and better-defined, task performed by developers are more likely to be routinized and standardized.

Routinization of work is a natural outcome of capitalist division of labor and its principles in workplace. At planning process, rational criteria of task assignment suggest optimal allocation of resources. Managers firstly look over available labor power in company. They evaluate workers in regards of who knows what and in how short time span particular worker can finish given task. Therefore, to save from time, divided and fragmented work parts are assigned to the people who have done similar tasks before. That means, same kind of tasks are started to be given to same people inevitably. Statements of Murat in this sense exemplify how working for a corporate company for a long time thwarts self improvement.

> Şu an oturmuş bir şey üzerinden devam ediyorum. Tabi katkıda bulunduğum noktalar var ama iş nasıl gelir, ne yapılır, az çok belli. Şunu da söyleyebilirim, iki seneyi geçti burada çalışmaya başlayalı, son bir birbuçuk senedir teknik anlamda yeni hiç bir şey öğrenmedim ben. Sadece şöyle daha iyi yaklaşılırmış, daha iyi çözülürmüş diyorum. Aa bak bu da böyle yapılıyormuş diye şaşırmıyorum. [Murat, 28][83]

Doing same part of work may set barrier against to the workers' necessity of skill upgrading. This situation increases the skill related tension experienced by

[83] Currently, I am working on an project which has been going on for a long while. Of course, I do contribute to this project but the way you work, how to do it, these are all pre-determined. But I can tell this, I have been working here for more then 2 years now, technically speaking, I haven't learned anything new for the last 1,5 years. I can see new approaches and solutions but I am not surprised with the techniques I come across. [Murat, 28]

developers. Routinization of work and hence not being able to renew skills afflict developers. Eralp for example seems to quit his job for another in recent past for similar reasons.

> Yapacak iş arıyorsun. Bana iş verin desen bile, ellerinde iş yoksa sana veremiyorlar. Bu da insanı işten, firmadan soğutuyor. Bu hiç güzel bir şey değil abi. Sen eğer bir iki sene sonra daha ciddi bir pozisyonda başka yerde çalışmak istiyorsan, sana BI'da 4 sene ne yaptın diye sorabilirler. Ben 4 sene boyunca kod aldım, script yazdım dersen bu hiç hoş değil.[Eralp, 29][84]

Correspondingly, software developers who work for a company for so long may encounter with the situations in which the skills they have, do not lead the way of job security. Cihan, one of the respondents, is a good example for this situation: the distinctive expertise he has on the medical software has not offered any benefit in the sense of job security.

> O yüzden kaygım büyük. Evet çok güzel, dünyada 3-4 kişinin yaptığı işi yapıyordum, ama şu an hiç bir kıymeti yok, kimse açısından kıymeti yok. Benim ne kadar iyi olduğumun hiç bir kıymeti yok. para etmiyorsa kıymeti yok. Durum bu. [Cihan, 29][85]

As Cihan also articulates, in software development sector, obtaining extensive expertise on a subject or working in a single company is not as advantageous as it is thought to be. That is to say, software developers who work for the same job throughout the years may fall short in catching up with the new technologies and

[84] You start looking for new works to do. Even if you ask the company to assign you new works, if you don't have any, they can't assign you anything. You get disinclined of the job after a while. This is not good at all. If you apply for a job in senior position in 1-2 years, they would ask you "what have you done in the BI in these 4 years?" If you say "I have received codes and written scripts", this would not be convenient, satisfying answer. [Eralp, 29]

[85] That's why I have huge concerns. Yes, that's great, I do a work only 3-4 people can do in the world. But it is not of value, no one appreciates it. It doesn't matter how good I am at my job. If it doesn't make any profit, than it is not valuable. This is how it is. [Cihan, 29]

new projects. For that matter, software developers mostly prefer switching jobs and acquiring multiple expertise, instead of working for a company for long time and deepening expertise on a specific field of software development. As an example of this tendency, having changed many jobs in 10 years is obviously an implied must according to Cemil's declarations:

> Ben ilk yıllarda güvenlik alanında uzmanlaşıyım dedim, bilişim güvenliği sektöründe. Baktım çok da iç açıcı bir alan değil. Bir yerlere ulaşamıyacam, kariyer planı yapabileceğim bir alan değil. Sonrasında da şunu farkettim: bir alanda uzmanlaştığında kariyer planı yapamıyorsun[m1] . Zaten o alanla ilgili, yani bir uzmanlık gerektiren firmalar azdır. Türkiye'de genelde ihale usülü işler alındığı için, general purpose amaçlı projeler daha öne çıkıyor. Ben de öyle yaptım, proje mühendisi oldum, öyle diyim. Ama 10 yıl çalıştığım için bir uzmanlığım var. İlk yılımdan beri uğraştığım, sürdürdüğüm bir uzmanlığım yok. [Cemil, 34][86]

In this sense, skills associated with conceptual and managerial parts of software production like experience on software development life cycle are distinguished among others. These kind of skills are less likely to lose their market validity, therefore, having those is seen as more secure option by workers for job continuity. Therefore, software developers tend to acquire skills that answer more common needs and transferable from one company to another.

> Daha kıdemli bir pozisyon bulurken, adam senden kullandığın teknoloji konusunda, proje yürütme konusunda kendini yetiştirmiş olmanı bekliyor, yazılım geliştirme proje döngüsünü ne kadar iyi biliyorsan o kadar avantajlı oluyorsun. Mülakata giren adam'ın, firmanın senden beklediği bu. Ben de ne

[86] I tried to specialize on security in my first years, on software security. Later, I didn't like it. It is not a subject on which I can build my career. Later I realized this: if you specialize on one topic, you cannot have a career plan. There are a few companies asking for specialization. Since in Turkey there is a bidding system, companies start projects with general purpose. I chose the same way. I have become a project engineer. But I have an expertise since I have been working for 10 years. But I don't have expertise on which I have been working since the very first day. [Cemil, 34]

> kadar farklı teknoloji bilirsem, bu döngüyü ne kadar bilirsem kendim için o kadar faydalı olur diye düşünüyorum. [Eralp, 29][87]

In order to acquire transferable skills, software developers need to work both in different projects as far as possible and for different range of companies. Multifirm careers, experience on different projects serve as a strategy of achieving transferrable skills. This situation, as expected, brings about changing jobs frequently which is a strategy for job security.

> Ama noluyor, bir konu üzerine ya da bir projede 5 sene çalışmak da aslında az rastlanır bişi bizim sektörde. [Murat][88]

Related to this situation, I should put forward that projects being executed at workplace hold crucial importance since they are the most prominent resource of skill renewment according to software workers. Thus, selection of project and workplace, in the sense of content of work is the key point of staying productive and competitive in labor market. Words of Emrah exemplifies that content of work and from this point, skills to be developed should be taken as a top priority at developers' career plans, even before wages.

> Aslında bu sektörün en büyük parametrelerinden birisi kendini ne kadar geliştirdiğin, maaşına da bakmayacaksın. Bir şirkete girdiğinde kendine ne

[87] If you want to be employed in better-ranked positions, companies expect you to have been trained yourself about the technology you use. The more you know about software development cycle, the more advantaged you are. This is what company expects you to do. That's why, I think that I should learn as much various technology and softare development cycle as I can. [Eralp, 29]

[88] It is also hard to find someone who has worked 5 years on a topic in this sector. People usually change their topics. Once the company doesn't need it anymore, it changes the topic.

> kadar çok şey katacaksın, bunlara bakman gerekiyor, burası önemli. Bunu aşamamışsan bu sektörde çalışmak çok zor. [Emrah, 25][89]

For all these reasons, having trouble in self-improvement is a widespread and very critical problem among the software developers. According to a survey conducted by the BMO (Chamber of Computer Engineers)[90] in 2009, 40% of the participants define not being able to upgrade skills as a problem they encounter in the workplace. In such cases that software developers' demand to be assigned to different tasks are not responded by companies, it is very usual for developers, specifically for younger ones, to quit their jobs and seek another.

> Kendini yenilemek istediği halde, yenileyemediği için, bundan da mutsuzluk duyuyorsa, bana ya yeni bir iş ver, ya da ben gidecem diyor. Başka bir iş verebilirsen kalıyor, ama veremezsen de başka iş veren birisine gidiyor. [Yüksel, 40][91]

> Maaş tatmin ediyorsa ve işten ayrılıyorsa yazılımcı, bu mesleki tatminsizliktir, yaptığı işi beğenmiyordur. Çünkü kimse aynı kodu yazmak istemez. Biraz da şey kaygısı var, iş kaygısı... Çünkü özel sektördesin, çalıştığın firma bugün var yarın yok, senelerce PHP yazarsın, çıktığında başka bir yere girmek istediğinde, PHP badireler atlatmıştır, yeni teknolojiler çıkmıştır, hiç birisini bilmiyorsan, eski teknolojiyi kullanan yer de kalmamışsa iş bulamazsın.[Emrah, 25][92]

---

[89] In fact, one of the most important criteria in this sector is the self-improvement, regardless of the wage. Once you get a job, you should care about how much you can contribute to your professional development, this is important. If you can't get over it, it is extremely difficult to work in this sector. [Emrah, 25]

[90] http://www.bimo.org.tr/wp-content/uploads/2011/08/AnketSonucRaporuv2.pdf

[91] Think about it. Someone wants to refresh herself and her talents, but she cannot no matter how much she wants. She feels unhappy. She tells you "either assign me a new job or I am quitting". If you can assign a new job, she stays, otherwise, she finds another new job. [Yüksel, 40]

[92] If the salary is satisfying for a software developer and she quits the job, this is professional dissatisfaction. Because, no one wants to write the same code over and over

In this respect, as high rate of job turnovers proves that (Benner, 2002, 45), switching from one project to another appears as a strategy of staying competitive and employable in labour market. Suggestion which Eralp took from his friends is very explanatory in this sense:

> Amerika'da oda arkadaşlarım vardı, 2 tane Japonla kalıyordum. Onların bana bir tavsiyesi vardı, buna uymaya çalışıyorum hep. Bir firmaya girdikten sonra, o firmada yapacağın işin, o firmadan kapacağın teknolojilerin ömrü maksimum 2 senedir ya da 1 senedir diyorlardı. Bu sürenin sonunda yeni orjinal projeler sağlamıyorlarsa, sen de kendini geliştiremiyorsan o firmadaki çalışma süren dolmuştur diyorlardı. Onun da yahoo'dan önce çalıştığı birçok firma vardı, wallmart'ta 6 ay çalışmış, yahoo'da 1 senedir çalışıyordu, ben gittikten sonra yahoo'dan da ayrıldığını duydum. Kendini geliştirmek için yapıyorlar yani. [Eralp, 29][93]

All things considered, future uncertainty creates the concerns about job insecurity. These concerns lead the way to self-exploitative practices of software developers. In an environment where long term job security is most probably dependent upon the skills renewal and self-improvement, software developers feel the urge to keep up with the market demands. However, these practices turn out to be self-exploitative in two respects, firstly, with the pressure of self-improvement, they feel the pressure of high performance for their own sake rather than their employers (Ehrenstein,

---

again. Finding job is another concern. You work in private sector, the company you work for might not survive in the future. If you might have been coding PHP for long time and once you start looking for another job, you come to see that PHP has been outmoded, there are new technologies now. If you are not familiar with any of them, you cannot find a new job. [Emrah, 25]

[93] I had two Japanese roommates in the US, both were IT specialist. They gave me an advice which I always try to keep in mind: "After you start a job in a firm, the lifetime of the job or of the technologies you will learn in that firm is no longer than 1-2 years. If, at the end of this time, the firm doesn't offer you anything new, if you cannot improve yourself, then you shouldn't stay in that firm." Before Yahoo, he worked for Wallmart for 6 months, he had been working for Yahoo for 1 year. After I left, he quit Yahoo, too. They do this to improve themselves. [Eralp, 29]

2006 cited in Papadoupoulos et al., 2008). Therefore, software developers comply with the market demands by developing certain strategies. Secondly, workers switch jobs quite frequently to be involved in as many different projects as possible. That is to say, workers commonly embrace strategies which are showing market-oriented and individualistic characteristics.

However, these market oriented strategies bolster the relations that led to insecure employment relations, rather than to bring permanent solutions to the workers' problems. They lead the way through cheapening of labor power and further insecurity. In this way, because, labor can be intensified with insecure and short-term employment relations, so productivity growth can be achieved. However, productivity growth turn back to workers as amplified precarious and insecure employment relations. Relation between intensification of labor/productivity growth and job insecurity, because, is a mutual one as Magdoff states:

> Regardless of the potential for workers to share some of the income generated by increased productivity, more efficient production creates problems for labor. With greater labor productivity—whether produced by new machines or more effective management techniques for controlling labor ("doing more with less," as they say nowadays)—fewer or less-skilled workers are needed for a given level of output. (Magdoff, 2004, 24)

Thus, it can be said that individualistic and market-oriented solutions taken as remedy to current insecure employment relations by developers have become very part of the reproduction of the same conditions. That is to say, these strategies developed and used by the workers themselves reproduce the current production relations by contributing the competition in labor market and reproduction of labor power while ostensibly/seemingly securing the future for the employees.

CHAPTER V

CONCLUSION

Software applications are one of the most critical elements of information and communication technologies [ICT's] widely used today. They have been permeated into all economic activities through these technologies as well as social activities. As economic entities themselves, the focus on software applications have been generally dealt with their consumption and impacts of their utilization. However, very few studies in Turkey has gone into the detail of the how and through which production relations software development is conducted. This thesis, in this respect, is an attempt firstly to contribute to the Turkish literature on information and communication technologies, particularly software development, by focusing on the production of software applications within a general context of capitalist production relations.

In the introduction chapter, objective of the study, fundamental characteristics of field research and brief theoretical background have been presented, through which the collected information of case study is developed. Different notions, propounded to describe the era and transformation observed in economy and society after the 1970's were presented first. In this regard, the contrast between theoretical approaches which tend to emphasize the changes and to propose that new era has been emerged; and which stick with continuities and which choose to evaluate transformation under the light of these continuities were mentioned. Among these approaches, flexible accumulation regime and its strategies are accepted as the primary structure giving a shape today's relations of production, hence the

capital/labor conflict in capitalism's historical specificity. Definition of flexible production organization and its effects on labor market were briefly discussed. In this regard, the employment relations which are generally defined under the precariousness notion which affects lots of industries including high-skill jobs were taken into consideration. Labor process theory, upon which this thesis is remarkably based, was dealt with its prominent notions with references to Braverman and Burawoy. I concluded introduction chapter by giving outlines of chapter and the details of the field study with its methodology.

In chapter 2, technical organization of software development and particular means of production were interrogated with the help of findings from field study. It can be proposed that worker's skills, experience, and talent are largely individual features, while "the complexity of tools and tasks reflect the social relations of the workplace and of the society which created the workplace" (Kraft and Dubnoff, 1986). Therefore, specific characteristics of the technical organization of software development were analyzed. Based upon our field study, it was argued that means of production of software development are used as counterparts of the capital for the specific technical organization of the production process.

It was observed that incessant transformation of means of production has shown tendencies towards standardization, fragmentation and automation of production within short history of software development. As a result, discretion of workers over their work seems to be decreased whereas managements' control over it has been expanded. By this means, skill requirements needed for software development has been relatively decreased. That is to say, software development as a practice has become possible for a wider spectrum of people.

As a result of this specific organization of software development, workers seemed to have become more disposable and easily replaceable for capital in the technical

sense. It was asserted that, this situation exacerbates vulnerability of software developers against capital, as indicated specifically by older software developers. This was apparent in their statements implicitly reflecting discontent about the outcomes of standardization and simplification of programming. Their self-respect in the sense of work identity seemed to be damaged as they got aware how their work experience is getting eroded in labor market due to this incessant transformation of production organization.

For younger developers, on the other hand, degradation of work was more urgent problem. Having less pressure about their future job security than older developers do, they seemed to be more concerned about their job satisfaction. In this sense, their statements were very precise about how their work tasks are routinized and simplified at their jobs. It can be proposed therefore, technical organization of software development which is significantly decisive on production relations were remarkably shaped in this industry in accordance with interests of capital, that is, capitalist accumulation.

In chapter 3, relations of control between management and software developers at workplace were delineated. It was argued that, because of workers' power which is emanated from their control over technical part of software development, control strategies based on consent are prevalently utilized by management in this sector. It was commonly observed that, relative autonomy in regards of technical decisions and working time arrangements is given to workers. However, this autonomy does not mean absolute freedom for workers, its limitations are carefully determined by management in a way to secure and increase accumulation of surplus labor value. Autonomy and freedom given to workers in this sense, seemed to be the freedom to meet the economic demands of the companies.

Direct control strategies based upon rules, regulations and organization structure of companies were also observed in software industry. It was argued that bureaucratic and disciplinary control strategies still prevail even vertical and rigid production organization structure of companies has given a way to horizontal and flexible production organization. Based on these findings, it was asserted that relations of control at workplace either based on coercion or consent, have been formed around imperatives of surplus value accumulation.

In chapter 4, the software developers' self-developmental concerns, its connection with skills and factors that led to job insecurity among software developers were analyzed. It was asserted that, software developers are significantly afflicted by pressure of uncertain future in the sense of job security. Flexible organization structure of production and short-term employment relations make it difficult for software workers to predict their own future within software industry. As stated by respondents, what will happen even in near future is an enigma. Having considered that age more likely bring disadvantages in this sector rather than advantages, pressure of job insecurity upon workers got worsened over time. Aware of the fact that promotion to managerial positions is not possible for every one of them, workers are worried about till what age they would be employed as technical workers in software industry. Parallel to these, it was seen that the future plans and expectations of the interviewees are generally uncertain and unstable.

It was observed that software developers are generally bound up with individualistic and market-oriented strategies in order to deal with job insecurity and future uncertainty. Renewal of skills is the most common strategy employed by workers in order to stay competitive and employable in labor market. Under these conditions, majority of software developers seemed to be subordinated to capitalist production relations and participate to the reproduction of its conditions.

It should be clarified that, what software developers experienced in the sense of precariousness and job insecurity is different from experiences of workers from lower classes. For low-skilled workers, future uncertainty is not a crucial problem as it is for software developers. Being aware of that they are disposable and replaceable in labor market, future is perceived by 'blue collar' workers as something out of their power and control (Erdoğan, 2011, 100). They rarely experience job insecurity or unemployment in terms of self-respect. However, 'white collar' workers, as subjects who believe that their future is in their control, tended to see job insecurity and other work-related problems as outcomes of their decisions (Erdoğan, 2011, 91-92). Similarly, it has observed in this field study that software developers engage with their problems as their own and believe they can and should overcome those on their own. Thus, they experienced future uncertainty and job insecurity more likely as a problem of self-respect in the sense of work identity. In this regard, family background of workers might be significantly determinative aspect. Respondents of this research mostly come from middle class families and have social and cultural values of this class. Therefore, as asserted by Erdoğan (2011, 92), encountering with that they are also disposable and replaceable parts of production, despite of their high skills and qualifications, may have exacerbated complications of the respectability problem which they experience.

Despite the common problems and complaints related to working conditions, it is hard to mention that there is a collective political behavior amongst workers in this industry. It is commonly observed that software workers rarely identify themselves through collective identity. In interviews, majority of respondents stated that software developers need for worker union or any other kind of worker organization which protect their rights. However, they were very pesimistic as to whether such organization could ever be established or could protect their rights under these conditions. High competition in labor market, short-term employment relations and their relatively privileged position in general production organization seem to push

software workers into individualized feelings in which workers compete with each other for themselves rather than collaborate for their common rights. Nevertheless, there can be mentioned of a potential which may lead to strengthen collective identity and political movement among software developers. Solidarity networks established for different purposes like Bilişim Çalışanları Dayanışma Ağı [BİÇDA][94], Kadın Yazılımcılar[95], İstanbul Hackerspace[96] which seem to supporter of Free Software movement may provide a common basis in which politics of production and collective political movement among workers can originate.

[94] http://bilisimcalisanlari.org/

[95] http://www.kadinyazilimci.com/

[96] https://istanbulhs.org/wiki/

REFERENCES

Aglietta, Michel. *A Theory of Capitalist Regulation: The US Experience*. Transl. by David Fernbach. London: NLB, 1979.

Armağan, A. C., "Bilişim Sektöründe Emeğin Statüsü: Türkiye'de Bilişim Çalışanlarının Vasıfları Üzerine Bir Araştırma", PhD diss., İstanbul Üniversitesi, 2012.

Barrett, R. "Laboring Under an Illusion ? The Labor Process of Software Development in the Australian Information Industry." *New Technology, Work and Employment* 16, no. 1 (2001): 18-33.

Barrett, R. "Working at Webboyz: An Analysis of Control Over the Software Development Labour Process." *Sociology* 38, no. 4 (2004): 777-94.

Barrett, R. "Managing the software development labour Process Direct control, time and technical autonomy" In *Management, Labour Process and Software Development Reality Bytes*, edited by Rowena Barrett, 65-89. London: Routledge, 2005.

Bauwens, Michel. "The Political Economy of Peer Production." 2005. Accessed April 4, 2015. www.ctheory.com.

Beirne, M., Ramsey, H. and Panteli, A. "Softening the System: Conflict and Contradiction in Computing Work." In *Workplaces of the Future*, edited by Paul Thompson and Chris Warhust. Basingstoke: Macmillan Business, 1998.

Bell, D. *The Coming of Post-Industrial Society : A Venture In Social Fore-Castin.* New York: Basic Books., 1973.

Benner, Chris. *Work in the New Economy: Flexible Labor Markets in Silicon Valley*. Oxford, UK: Malden, MA :, 2002.

Berry, David M. *The Philosophy of Software: Code and Mediation in the Digital Age*. Basingstoke, Hampshire: Palgrave Macmillan, 2011.

Bora, A., Bora,T., Erdoğan, N. and Üstün, İ. *Boşuna mı Okuduk? Türkiye'de Beyaz Yakalı İşsizliği*. İstanbul: İletişim Yayınları. 2011.

Bora, T. and Erdoğan, N. "Cüppenin, Kılıcın ve Kalemin Mahcup Yoksulları: Yeni Kapitalizm, Yeni İşsizlik ve Beyaz Yakalılar." In *Boşuna mı Okuduk? Türkiye'de Beyaz Yakalı İşsizliği* edited by T. Bora, A. Bora, N. Erdoğan, & İ. Üstün, İstanbul: İletişim Yayınları, 2011.

Boyer, Robert. *The Regulation School: A Critical Introduction*. New York: Columbia University Press, 1990.

Braverman, Harry. *Labor and Monopoly Capital the Degradation of Work in the Twentieth Century*. 25th Anniversary ed. New York: Monthly Review Press, 1998.

Burawoy, M. "Toward a Marxist Theory of the Labor Process: Braverman and Beyond." *Politics & Society*, (1978), 247-312.

Burawoy, Michael. *Manufacturing Consent: Changes in the Labor Process under Monopoly Capitalism*. Chicago: University of Chicago Press, 1979.

Burawoy, Michael. *The Politics of Production: Factory Regimes under Capitalism and Socialism*. London: Verso, 1985.

Burchell, Brendan. "The Prevalence and Redistribution of Job Insecurity and Work Intensification." In *Job Insecurity and Work Intensification*, edited by Brendan Burchell, David Lapido, and Frank Wilkinson, by. London: Routledge, 2002.

Castells, Manuel. *The Rise of the Network Society*. Malden, Mass.: Blackwell Publishers, 1996.

Candeias, M. "Unmaking and Remaking of Class The "Impossıble" Precariat Between Fragmentatıon And Movement", 2007. In http://www.rosa-luxemburgclub.de/fileadmin/rls_uploads/pdfs/Policy_Paper/pp-3-09_Candeias.pdf

Candeias, M. “Double precarisation of labour and reproduction – Perspectives of expanded (re)appropriation”, 2009. In http://www.rosalux.de/fileadmin/wgdw_uploads/ Double_precarisation.pdf

Çerkezoğlu, A. and Göztepe, Ö. “Sınıfını Arayan Siyasetten Siyasetini Arayan Sınıfa”, in *Tekel Direnişinin Işığından Gelenekselden Yeniye İşçi Sınıfı Hareketi*, edited by Bulut, G., Nota Bene Yayınları, Ankara. 67-97, 2010.

Davenport, Thomas H., and Lawrence Prusak. "Working Knowledge: How Organizations Manage What They Know." *Ubiquity*, 2000.

Dubey, Abhijit and Wagle, Dilip. "Delivering Software as a Service." *The Mckinsey Quarterly, Web Exclusive*. 2007.

Elger, T. "Valorisation and 'Deskilling': A Critique of Braverman." *Capital & Class*, 1979, 58-99.

Erdoğan, N. “Sancılı Dil, Hadım Edilen Kendilik ve Aşınan Karakter: Beyaz Yakalı İşsizliğine Dair Notlar ” In *Boşuna mı Okuduk? Türkiye'de Beyaz Yakalı İşsizliği* edited by T. Bora, A. Bora, N. Erdoğan, & İ. Üstün, İstanbul: İletişim Yayınları. 2011.

Flores, Fernando, and John Gray. *Entrepreneurship and the Wired Life: Work in the Wake of Careers*. London: Demos, 2000.

Foster, John B. "Introduction" to *Labor Monopoly and Capital The Degradation of Work in the Twentieth Century*, by Harry Braverman. New York: Monthly Review Press, 1998.

Friedman, Andrew L. *Industry and Labour: Class Struggle at Work and Monopoly Capitalism*. London: Macmillan, 1977.

Gordon, David M., and Richard Edwards. *Segmented Work, Divided Workers: The Historical Transformation of Labor in the United States*. Cambridge: Cambridge University Press, 1983.

Greenbaum, J. "The Times They Are a Changing: Dividing and Re-Combining Labour through Computer Systems." In *Workplaces of the Future*, edited by Paul Thompson and Chris Warhust,. Basingstoke: Macmillan Business, 1998.

Harvey, David. *The Condition of Postmodernity: An Enquiry into the Origins of Cultural Change*. Oxford [England: Blackwell, 1989.

İliç, E. "Türkiye'de Bilişim İşçileri: Vasıf ve Denetim Üzerine Bir Çalışma", Master Thesis, Ankara Üniversitesi, 2011

Jessop, Bob. *The Future of the Capitalist State*. Malden, MA: Polity, 2003.

Kitay, Jim. "The Labour Process: Still Stuck? Still a Perspective? Still Useful?" *Electronic Journal of Radical Organiation Theory* 3, no. 1, 1997. http://www.management.ac.nz/ejrot/vol3_1/kitay.pdf.

Kodalak, M.C. "Personal Consequences of Work Under the 'New Economy': the Case of METU-Technopolis", Master Thesis, Middle East Technical University, 2007

Kraft, Philip. *Programmers and Managers: The Routinization of Computer Programming in the United States*. New York: Springer-Verlag, 1977.

Kraft, P. "The Routinizing of Computer Programming." *Work and Occupations*, 1979, 139-155.

Kraft, Philip, and Steven Dubnoff. "Job Content, Fragmentation, and Control in Computer Software Work." *Industrial Relations*, 1986, 184-96.

Landes, David S. *The Unbound Prometheus: Technological Change and Industrial Development in Western Europe from 1750 to the Present*. London: Cambridge U.P., 1969.

Littler, C. R., and P. Innes. "Downsizing and Deknowledging the Firm." *Work, Employment & Society*, 2003, 73-100.

Littler, Craig R., and Graeme Salaman. "Bravermania and Beyond: Recent Theories of the Labour Process." *Sociology Sociology*, 1982, 251-69.

Magdoff, Fred, and Harry Magdoff. "Disposable Workers: Today's Reserve Army of Labor." *Monthly Review Mon. Rev.*, 2004, 18.

Marx, Karl. *Capital: A Critique of Political Economy*. Harmondsworth: Penguin Books, 1992.

Naisbitt, John. *Megatrends: Ten New Directions Transforming Our Lives*. New York, N.Y.: Warner Books, 1984.

Neilson, B., and N. Rossiter. "Precarity as a Political Concept, Or, Fordism as Exception." *Theory, Culture & Society*, 2008, 51-72.

Özdemir, Ali Murat, and Gamze Yücesan-Özdemir. *Sermayenin Adaleti: Türkiye'de Emek Ve Sosyal Politika*. 1. Baskı. ed. Kızılay, Ankara: Dipnot Yayınları, 2008.

Quintas, Paul. "Programmed Innovation? Trajectories of Change in Software Development." *Information Technology & People Info Technology & People*, 1994, 25-47.

Papadopoulos, Dimitris, Niamh Stephenson and Vassilis Tsianos. *Escape Routes Control and Subversion in the Twenty-first Century*. London: Pluto Press, 2008.

Perrolle, J. A. "Intellectual Assembly Lines: The Rationalization of Managerial, Professional, and Technical Work." *Social Science Computer Review*, 1984, 111-121.

Philipson, G. "A Short History of Software." In *Management, Labour Process and Software Development Reality Bytes*, edited by Rowena Barrett, 12-40. London: Routledge, 2005.

Piore, Michael J., and Charles F. Sabel. *The Second Industrial Divide: Possibilities for Prosperity*. New York: Basic Books, 1984.

Rodgers, G. and Rodgers, J. “Precarious jobs in labour market regulation: The growth of atypical employment in Western Europe”, *International Institute of Labour Studies*, 1989.

Sağıroğlu, S. “From Precarious Employment to Precarious Life: The Case of Non-appointed Teachers in Turkey”, Master Thesis, Middle East Technical University, 2013.

Scarbrough, H. *Knowledge Management: A Literature Review*. 1999.

Scott Hanselman." - Coder, Blogger, Teacher, Speaker, Author. htttp://www.hanselman.com, 2013.

Sennett, Richard. *The Culture of the New Capitalism*. New Haven: Yale University Press, 2006.

Scott, Allen John. *Flexible Production Systems and Regional Development: The Rise of New Industrial Spaces in North America and Western Europe*. Toronto: Centre for Urban and Community Studies, University of Toronto, 1988.

Standing, Guy. *The Precariat the New Dangerous Class*. London: Bloomsbury Academic, 2011.

Thompson, Paul. "Disconnected Capitalism: Or Why Employers Can't Keep Their Side of the Bargain." *Work, Employment & Society*, 2003, 359-78.

Thompson, Paul, and Chris Warhust. "Introduction." In *Workplaces of the Future*. Basingstoke: Macmillan Business, 1998.

Turner, M., D. Budgen, and P. Brereton. "Turning Software into a Service." *Computer*: 38-44. 2003

Voss-Dahm, D. (2005) “Coming and going at will ? Working time organization in German IT companies” In Management, Labour Process and Software Development Reality Bytes, edited by Rowena Barrett, 109-129. London: Routledge

Wasko, M. Mclure, and S. Faraj. "“It Is What One Does”: Why People Participate and Help Others in Electronic Communities of Practice." *The Journal of Strategic Information Systems*, 2000, 155-73.

Webster, Frank. *Theories of the Information Society*. 2nd ed. London: Routledge, 2002.

Yücesan-Özdemir, G. “Emek süreci teorisi ve Türkiye’de Emek Süreci Çalışmalar Üzerine Bir Değerlendirme”, Küreselleşme Emek, 2002.

Yücesan-Özdemir, G. *İnatçı Köstebek: Çağrı Merkezlerinde Gençlik, Sınıf ve Direniş* . İstanbul: Yordam Kitap, 2014.

APPENDIX A: THE INFORMATION OF INTERVIEWEES

TABLE 1

İsim	Yaş	Şehir	Öğrenim durumu	Mezun olunan bölüm (lisans)	Mezun olunan üniversite	Çalışılan firma ölçeği
Metin	33	Ankara	Doktora (mezun)	Elektrik & Elektronik Müh.	ODTÜ	Orta
Leyla	25	Ankara	Y.Lisans	Bilgisayar Müh.	ODTÜ	Orta
Murat	28	Ankara	Lisans (mezun)	Bilgisayar Müh.	ODTÜ	Büyük
Ceren	25	Ankara	Y.Lisans	Uzay ve Havacılık Müh.	ODTÜ	Orta
Eralp	29	İstanbul	Y.Lisans (mezun)	Bilgisayar Müh.	Bilkent Ü.	Orta
Emrah	25	İstanbul	Lisans (mezun)	Kimya Müh.	Atatürk Ü.	Orta
Elif	37	İstanbul	Y.Lisans (mezun)	Kimya Müh.	ODTÜ	Küçük
Gökhan	44	Ankara	Doktora (mezun)	Elektrik & Elektronik Müh.	ODTÜ	Orta
Ezgi	24	İstanbul	Lisans (mezun)	Matematik müh.	Yıldız Teknik Ü.	Büyük
Cemil	34	Ankara	Y.Lisans (mezun)	Bilgisayar Müh.	ODTÜ	Küçük

TABLE 1 (continued)

Semih	44	İstanbul-Ankara	Lisans (mezun)	Fizik	ODTÜ	Büyük
Sevil	40	İstanbul	Lisans (mezun)	Elektrik & Elektronik Müh.	İTÜ	Küçük
Kutay	25	Ankara	Lisans	Bilgisayar Müh.	Ankara Ü.	Küçük
Kazım	41	Ankara	Lisans (mezun)	Matematik	ODTÜ	Büyük
Erdem	23	İstanbul	Lisans (mezun)	Bilgisayar Müh.	TOBB Ü.	Orta
Çağdaş	32	İstanbul	Lisans (mezun)	Matematik müh.	Yıldız Teknik Ü.	Büyük
Yüksel	40	Ankara	Y.Lisans (mezun)	İstatistik	ODTÜ	Büyük
Hakan	26	Ankara	Y.Lisans	Bilgisayar Müh.	ODTÜ	Büyük
Cihan	29	Ankara	Doktora	BÖTE	ODTÜ	Küçük
Barış	24	İstanbul	Lisans	Bilgisayar müh.	Bilgi Ü.	Freelance
Fırat	30	Ankara	Lisans (mezun)	Bilgisayar Müh.	Atılım Ü	Büyük

APPENDIX B: INTERVIEW PROTOCOL

-Temel Bilgiler
- -yaş
- -cinsiyet
- -eğitim formasyonu
- -çalışılan şehir
- -istihdam biçimi
- -çalışılan şirketin büyüklüğü
- -Aile, doğum yeri, nerede büyüdüğü, yaşı, medeni durumu,
- -Eğitim, mezun olunan okullar, üniversite

-Çalışma Koşulları
- -kaç senedir sektörde çalıştığı
- -şu an çalıştığı iş yerinde kaç senedir çalışıyor olduğu
- -şirketin büyüklüğü
- -şirketin faaliyet gösterdiği konular
- -şirket yapısı
- -maaşlar
- -şirket içi pozisyon
- -yönetici/patron
  - -tabiiyet ilişkileri
  - -çalışanların gösterdiği aidiyet
  - -yönetimle ya da patronla yaşanan problem/sorun

-Ürün/Yazılım
- -yazılım nerelerde kullanılıyor?
- -kullanıldığı yerlere nasıl etkiyor, fayda sağlıyor ?

-Üretim Süreci
- -üretim örgütlenmesi,
- -yazılım nasıl bir organizasyon içerisinde geliştiriliyor
- -uzmanlık alanı
- -kullanılan standartlar, şemalar, organizasyonlar
- -zaman örgütlenmesi
- -mekan örgütlenmesi
- -teknik konularda yaşanan sorunlar
- -teknik sorunları/problemleri idare süreçleri
- -iş tatmini, creative ? challenging ? routine ?
- -denetleme mekanizmaları

-Vasıfların idaresi, Gelecek planları

-kendini geliştirmek ne anlama geliyor

-vasıflarını neden yeniliyorlar ?

-vasıflarını neye göre yeniliyorlar

- geleceğe yönelik planlar

- kendi işini kurmak

APPENDIX C: QUESTIONS OF SURVEY

1) yaşınız ?

2) cinsiyetiniz
askerlik durumunuz ?

3) medeni haliniz

4) hangi şehirde yaşıyorsunuz?

5) eğitim durumunuz (mezun olunan)
> hangi bölümprogram, hangi üniversite ?

6) eğitim durumunuz (varsa halihazırda devam edilen)
> yüksek lisans, doktora programı, sertifika programları?

6) devam edecek olsaydınız, yüksek lisans eğitiminize hangi alanda devam etmek isterdiniz ?
6) değiştirebilecek olsaydınız, lisans ya da yüksek lisans eğitiminizi farklı bir alandan seçmek ister miydiniz? Evetse, hangi alanda ?

7) bildiğiniz yabancı diller?

8) aşağıdaki çalışma koşullarından hangisinde yer alıyorsunuz ?

özel bir şirkette tam zamanlı sözleşmeli olarak çalışıyorum
özel bir şirkette tam zamanlı kadrolu olarak çalışıyorum
özel bir şirkette yarı zamanlı çalışıyorum.
özel bir şirkette proje bazlı çalışıyorum.
bir kamu kuruluşunda çalışıyorum.
kamuözel ortaklığındaki bir kuruluşta çalışıyorum.
freelance/selfemployed/bağımsız çalışıyorum.
şirket ortağı/sahibiyim.
şu anda çalışmıyorum.
diğer(?)

9) Durumunuza uygun olan seçeneği işaretleyiniz.

- Yazılım konusunda bağımsız faaliyet gösteren bir şirkette çalışıyorum.
- Farklı sektörde faaliyet gösteren bir kurumun yazılım ihtiyaçlarının karşılandığı departmanda/bölümde çalışıyorum.

10) iş yerinizde yazılım üzerine birlikte çalıştığınız yazılımcı/bilgisayar mühendisi/çalışan sayısı ?[ö,k]

11) ne kadar süredir şu anki iş yerinde çalışıyorsunuz ? [ö,k]
11) son iş yerinizde ne kadar süre çalışmıştınız ?[i]
11) ne kadar süredir bağımsız çalışıyorsunuz ?[f]

12) toplamda ne kadar süredir bilişim/yazılım sektöründe çalışıyorsunuz/çalıştınız?[g][i]

13) bugüne kadar kaç iş yerinde çalıştınız ? [g][i]

14) şimdiye kadarki iş değişikliklerinizin (işten ayrılma ya da çıkarılma) sebep veya sebepleri nelerdi ?[ö,k,i][f]

aldığım ücret yeterli/adil değildi.
çalışma saatleri yorucu ve yıpratıcıydı.
yöneticilerimle geçinemiyordum.
iş arkadaşlarımla geçinemiyordum.
kendimi mesleki anlamda geliştiremiyordum.
iş yerim mesleki pozisyon/ünvan olarak yükselme imkanı sunmuyordu.
şirket içindeki iş ve ücret dağılımı adaletsizdi.
daha güvenceli koşullarda çalışmak istedim.
daha iyi koşullara sahip bir iş teklifi aldım.
çalışmam için alındığım proje sona erdi.
diğer

15) şimdiye kadar hiç işsiz kaldınız mı ? [g]

hayır.
evet, bir kere, 2 aydan kısa bir dönem için işsiz kaldım.
evet, bir kaç kez, 2 aydan kısa süreler için işsiz kaldım.
evet, bir kere, 2 aydan uzun bir dönem için işsiz kaldım.
evet, bir kaç kez, 2 aydan uzun süreler için işsiz kaldım.
evet, kısa ve uzun süreler için iş bulamadığım oldu.

16) yeterlilik gösterdiğiniz uzmanlık alanları/konuları nelerdir?[g]

Veritabanı
Kodlama
Bilgi Güvenliği
Ağ
Sistem Programcılığı
Sistem Analizi
Sistem Destek
Sistem Tasarımı
Web Tasarımı
Web Destek
Test
Yazılım Kaliteciliği
Proje Yöneticiliği
Satış Öncesi Analistliği
Denetmenlik
Danışmanlık
Pazarlama
Konfigürasyon
Mobil Uygulamalar
Diğer

17) bu uzmanlık alanlarından şimdiye kadar hangisi ya da hangilerinde çalıştınız/hangilerini kullandınız?[g]

Veritabanı
Kodlama
Bilgi Güvenliği
Ağ
Sistem Programcılığı
Sistem Analizi
Sistem Destek
Sistem Tasarımı
Web Tasarımı
Web Destek
Test
Yazılım Kaliteciliği
Proje Yöneticiliği
Satış Öncesi Analistliği
Denetmenlik
Danışmanlık

Pazarlama
Konfigürasyon
Mobil Uygulamalar
Diğer

18) çalışma hayatınız boyunca yazılım sektörü haricinde bir sektörde çalıştınız mı? evetse, ne kadar süre, hangi sektörde, hangi pozisyonda ? [g]

18) iş yerindeki ünvanınız ? [ö,k]

Yazılım Geliştirme Uzmanı
Yazılım Uzmanı
Yazılım Mühendisi
Yazılım Geliştirme Mühendisi
Kıdemli Yazılım Geliştirme Uzmanı
Kıdemli Yazılım Mühendisi
Kıdemli Yazılım Uzmanı
Yazılım Entegrasyon Yöneticisi
Yazılım Geliştirme Sorumlusu
Yazılım Analisti
Yazılım Mimarı
Test Mühendisi
Veri Analisti
Veritabanı Yöneticisi (DBA)
Sistem ve Ağ Uzmanı

19) maaşınız/kazandığınız para (brüt) aşağıdaki aralıklarından hangisine denk düşmektedir ? [g][i]

500-1500 tl
1500-2500 tl
2500-4000 tl
4000-7000 tl
7000-10000 tl
10000+tl

20) çalıştığınız projelerin ortalama hangi süre zarfında bitirilmeleri planlanıyor/bekleniyor ? [g][i]

21) çalıştığınız projelerin ortalama piyasa değerleri ne kadar ? [g][i]

21) haftalık çalışma saatiniz ? [g][i]

22) hangi aralıklarla ekstra mesaiye kalıyorsunuz ? [ö,k][i]

hiç kalmıyorum.
nadiren kalıyorum.
bazen kalıyorum.
sıklıkla kalıyorum.
çok sık kalıyorum.
kalıyorum ama bu dönemler süreklilik arzetmiyor, projelere ve teslim tarihlerine göre değişiyor.

23) evden çalışmak zorunda kaldığınız durumlar oluyor mu ? [ö,k][i]

hayır, olmuyor
nadiren oluyor.
sıklıkla oluyor.
çok sık oluyor.
bazen oluyor.

24) evde ya da iş yerinde, işiniz için harcadığınız fazladan mesai için ücretlendiriliyor musunuz? [ö,k] [i]

evet ücretlendiriliyorum.
hayır, ücretlendirilmiyorum.
harcadığım mesainin bir kısmı için ücretlendiriliyorum.

25) çalışma hayatınızı yazılım sektöründe devam ettirebilmek için mesleki bilgi ve becerilerinizi güncel tutmanız ya da geliştirmeniz gerektiğini düşünüyor musunuz ? [g]

26) mesleki bilgi ve becerilerinizi güncel tutmak ya da geliştirmek için neler yapıyorsunuz? [g]

akademik eğitimimi sürdürüyorum.
şirket içi mesleki eğitimlere katılıyorum.
şirket dışı seminer, kurs ve sertifika programlarına katılıyorum
kitap ve basılı yayınlardan yararlanıyorum.
internetteki kaynaklardan faydalanıyorum.
işte öğrendiklerim yeterli oluyor, fazladan çalışma ihtiyacı hissetmiyorum.

çeşitli sebeplerden bir şeyler yapmama fırsatım olmuyor/kalmıyor.
bir şey yapmıyorum.

27) işinizle ilgili konularda internetteki kaynaklara hangi sıklıklarla başvuruyorsunuz ?[g]

Hiç başvurmuyorum.
Nadiren
Bazen
Sıklıkla
Her zaman

28) bu kaynakları çalışma saatleriniz dışında da kullanıyor musunuz? [g]

Hayır, kullanmıyorum
Nadiren
Bazen
Sıklıkla
Her zaman

29) mesleğinizle alakalı konularda yürüttüğünüz blog ya da web siteniz var mı ? [g]
Varsa → ….

30) mesleğinizle ilgili konularda düzenli olarak hangi kaynakları kullanıyorsunuz/takip ediyorsunuz, hangilerinde içerik üretimine katkıda bulunuyorsunuz ? [g]

stackoverflow.com
github.com
quora.com/SoftwareEngineering
reddit.com/r/programming
codeproject.com
armut.com
theserverside.com
dzone.com
codinghorror.com
joelonsoftware.com
slashdot.com
theregister.co.uk
developerfusion.com
hanselman.com
www.r10.net/

Kişisel Blog??:….
Grup Blogları: …..
twitter hesapları…
tutorialspoint.com
coderanch.com/forums
java.dzone.com
mkyong.com
yazgelistir.com

APPENDIX D: AN EXAMPLE OF THE INTERVIEWS

Emrah Aslan, 25 yaşında, Atatürk Üniversitesi Kimya Mühendisliği Bölümü mezunu. Anne ev kadını, babası kendilerinin işlettiği lokantada aşçıymış, emekli olmuş.Emrah, İstanbul Fatih semtinde doğmuş, büyümüş. Yazılım sektöründe çalışmaya başlamasının üzerinden çok süre geçmediği söylenebilir. Deşifresini okuyacağınız mülakat'ı 2015'in Şubat ayında İstanbul'da, Üsküdar'daki bir kafe'de gerçekleştirdik.

Hüseyincan Eryılmaz [HE]: Ses kayıt cihazını açıyım izninle. Madem konu buradan açıldı, buradan devam edelim. Skala, angular.js ya da başka benzer teknolojiler sizin alana (web uygulamaları geliştirmeye) nasıl etki etti ?

Emrah Aslan [EA]: Konuştuklarımız web teknolojisi. Skalayı çok iyi bilmiyorum ama Angular.js web teknolojisi, front-end'de, ön tarafta çalışılan bir teknoloji. Neyi değiştirdi, takım çalışmasını daha bi kolay hale getirdi özellikle. Eskiden bütün işleri tek bir adam yapıyordu, web master denen bir adam vardı. Bir websitesinin hem tasarımını hem back-end'ini, business logicini, database işlerini falan hep bu adam yapıyordu. Şimdi bunlar ayrıldı. Front-end takımı çıktı, Back-end takımı çıktı, yazılım yapan takım ayrı oldu, görsel için çalışanlar ayrıldı. Bunlar da artık kadrolaşmaya başladılar. Front-end'de önceden bir adam çalışıyorsa artık bir işi 5 kişi yapmaya başladı. Böyle olunca framework gerekti, bir versiyonlama sistemi, iki framework, bir de işin daha derli toplu olması gerekiyor. Kodun daha derli toplu olması gerekiyor. Önceden herkes istediği yere istediği koda, orta yere dan diye dalıp burada bu çalışacak diyordu. Artık öyle değil. Hem bu teknolojiler seni daha temiz kod yazmaya zorluyor, öyle de olmalı. Çünkü bir adam bir şirkette senelerce çalışmıyor. Yarın gidecek, başka birisi gelecek, gidenin kodunu gördüğü zaman orada hangi mevzunun döndüğünü anlaması lazım. Bir bunun için, iki işe hız kazandırdı. Bir çok paket hazır geldi, bir çok toollar geldi, takım çantası gibi. Bu bahsettiğin teknolojiler de benzer özellikler getirdi. Websitesinin işleyişiyle ilgili yenilikler de getirdi de, onlara girmiyim. Ama Angular'ın böyle popüler olmasının sebeplerinden birisi de arkasında Google'ın olması, reklamını Google'un yapması. Projeye hiç katkıda bulunmayacak insanlar bile onu stallayıp, fork ediyor github'da, sırf Google onların yaptıkları işi görüp beğenebilir diye.

HE: Projeye destek olanların böyle bir motivasyonu var diyelim, Google niye böyle bir şey yapıyor, çıkarı ne bundan ?

EA: Google kendisi kullanıyor bu teknolojiyi, geliştirilmesi yaygınlaşması onun da işine geliyor. Bootstrap de öyle mesela, front-end teknolojisidir, twitter kendi kullanmak için geliştirdiği bu framework'ü şimdi açık kaynak olarak dağıtıyor. Web teknolojileri söz konusuysa hepsi öyle, kullandığı, geliştirdiği teknolojiyi sunuyor firmalar. Böyle olması bir yandan güzel, tek kanaldan yürümesi yani. Çünkü diğer türlü geriye yönelik desteğin olmayabiliyor. Mesela Jquery arkasında bir kuruluşun olduğu bir teknoloji değil, bir kişi geliştiriyor. Bu yüzden Jquery versiyonu 1.2'den 1.9'a geçtiği zaman patlıyor mesela. Daha önceden yazmış olduğum bir kod var mesela yeni sitede kullanacaksın, kullanamıyorsun. İşte Jquery 2'de işler değişiyor, yeniden gidip conflict ayarları yapman gerekiyor. Teknolojilerin arkasında kuruluşların olması, bir işi standardize ediyor.

HE: Konuşmanın başında yazılımcıların bir şirkette uzun süreler çalışmamasından bahsettin, neden öyle bir şey var ?

EA: Sektörle alakalı. Yazılımcı bugün çalışıyor, memnun, bir şeyler öğreniyor. Öğrendiği şeyleri yapmasına izin verilmiyor çalıştığı şirkette çünkü öyle işler gelmiyor. Yazılımcı yerini değiştiriyor, yerine de başkası geliyor.

HE: Neden ki ama? Maaşını almıyor mu, neden değiştiriyor işini ?

EA: Bu sektörde öyle, mesele maaş değil aslında. Ne kadar daha çok öğrenebilirim hastalığı var yazılım sektöründe. Bizde de var, bende var, ofiste çalışan arkadaşlarda da var. Herkes bir yandan web-development yaparken mobil teknolojilere bakıyorlar. Halbuki mobil şeyimiz, müşterimiz yok. İphone geliştirmeye bakıyoruz hepimiz.

HE: Ne zaman yapıyorsunuz bunu, işiniz zatem yoğun değil mi ?

EA: Evde yapıyoruz, bir yandan PDF'leri okuyoruz. Sürekli dökümantasyon peşindeyiz.

HE: Niye peki ?

EA: (gülüyor) Böyle bir dürtü, onu da bilmem gerekiyor şeyi oluyor. Daha acaip şeyler yapan adamlar var ...

HE: Bu sizin için iş değil mi sonuçta?

EA: İş değil tam olarak, bu yazılımcıyı mutlu eden bir şey. Sevdiği işi yapmak bu aslında ve para önemli değil. Çünkü bu sektörde PHP'ciler para için çalışır. Oranlarına bakıyorsun, bir sürü PHP geliştiricisi var, Python'dan daha fazla. Nasıl olur ki diyorsun, daha iyi bir teknoloji PHP'den nasıl daha az kullanılıyor. Bir bakıyorsun Github repolarına Python'da

katkıda bulunan daha fazla. Karşılıksız bir biçimde katkıda bulunuyor adam. Python betikleri geliştiriyor, yeni frameworkler geliştiriyor, Github'da dağıtıyor.

HE: Türkiye'de mi Dünya'da mı ?

EA: Dünya çapında. Pythona katkıda bulunanlar müşteriye yönelik ürün çıkartmıyor, para karşılığında iş yapmıyorlar. Sadece python'a katkıda bulunuyor. Bunu farkediyorsun mesela. Adam para kazanmıyor bundan. Böyle bir kültür de var. Unix ilk çıktığında ATNT bunun kodunu kapatacağım dediğinde, FSF çıkıyor bunu açık yapmalıyız diye. Linux powers çıkıyor sonra falan. Sonra Linux üzerine bir sürü fork çıkıyor, Debian çıkıyor, Ubuntu çıkıyor, Sentos çıkıyor. Bunlar başta standardize bile olmuyorlar, toplanıp command line komutlarımızı bari standardize edelim diyorlar.

HE: Niye standardizasyon gerekiyor ki o noktada?

EA: Kullanıcıya pazarlayamıyorsun. Adam debian kullanıyorsa senelerce onu kullanıyor. Ubuntu ihtiyacı olduğunda, atıyorum bir proje için ihtiyacı oldu, eğitimi bir maliyet. Onu öğrenmesi gerek.

HE: Biraz daha geriye dönersek, yazılımcıların iş değiştirmelerine sebep olan şey gelen işten sıkılmaları mı yani ?

EA: Gelen işin tatmin etmesi... Maaş tatmin ediyorsa ve işten ayrılıyorsa yazılımcı, bu mesleki tatminsizliktir, yaptığı işi beğenmiyordur. Çünkü kimse aynı kodu yazmak istemez. Biraz da şey kaygısı var, iş kaygısı... Çünkü özel sektördesin, çalıştığın firma bugün var yarın yok, senelerce PHP yazarsın, çıktığında başka bir yere girmek istediğinde, PHP badireler atlatmıştır, yeni teknolojiler çıkmıştır, hiç birisini bilmiyorsan, eski teknolojiyi kullanan yer de kalmamışsa iş bulamazsın. Ve bu Türkiye'de çok var. Özellikle Türkiye'den müşterilere çalışanlar için bu böyle. Kendi aralarında oldukları için, hani spor yaparken kendinden güçlüyle yaparsınki gelişesin, adamlar sürekli kendi içlerinde oldukları için herkes kendini Senior zannediyor. Türkiye'de Senior çok fazla var, ya şimdi öyle. Adam PHP biliyor, object oriented çok iyi biliyor, Linux'u tanıyor, web sunucusunu kuruyor, çalıştırıyor, A'dan Z'ye hepsini biliyor, ama bu bile Senior değildir. Dünya çapında zaten değildir de... Türkiye'de belki öyledir. Fakat bugün sektörde herkes Senior. İş kaygısının sebeplerinden biri de şu; Türkiye'deki işverenler şey... Sen hep az çok vizyonu olan firmalarla görüşmüşsün ama öyle adamlar var ki, 2 gün içerisinde şu iş yetişecek diye yazılımcı çalıştırıyorlar. 1500 lira para veriyorlar, dünyanın işini yaptırıyor. Herhangi bir framework kullanmıyorlar, bütün kodlar spagetti, adam gece gündüz yazıyor. Yani böyle insanlar da var, bu yüzden iş kaygısına düşüyorsun. Bu tür adamlara gidip de maaş beklentini söylesen, ben 1500'e neler neler yaptırıyorum diyor. Yaptığı işler de belli. Bir tane haber sitesi yaptırıyor, bir tane kurumsal web sitesi yaptırıyor.

HE: Kurumsal yaptırabiliyorlar yani … İlişkiler sayesinde falan mı ?

EA: Aynen. Hal böyle olunca, böyle bir sektörde tabi ki iş kaygın oluyor. Benim bir çok arkadaşım sürekli Stack Overflow'un kariyer sayfasından yurtdışı işler bekliyorlar. Location olan, vize işlemlerini şirketin yaptığı, yerleşmene yardımcı olacak firmalara bakıyorlar, Hollanda'da, Almanya'da, San Fransisco'da, sürekli iş takip ediyorlar.

HE: Bu arkadaşlarının Stack Overflowdaki reputation puanları yüksek mi peki ?

EA: Aslında profile çok bakmıyor onlar. Kendileri mülakat yapıyorlar skype üzerinden görüşüyorlar. Blog sayfan olmalı mutlaka. Elbette SO'ya da bakıyorlar da, tek kriter o değil. Skype ile mülakat yapıyorlar, ingilizce yazmış olduğun bir blog olmalı çalıştığın alanla ilgili.

HE: Peki bu arkadaşların blog tutuyorlar mı?

EA: Tabi yapıyorlar. Yazmış olduğu yazılar vardı, bir gün içerisinde blog'u açıp, koydu oraya. Normalde blog tutmuyor ama kenara aldığı notları blog yazısı haline getirip, bir gecede koyup, mülakatta bunları yazdım diyebiliyorlar.

HE: Yaptığı işlerle mi alakalı yoksa ayrıca takip ettiklerini mi yazıyor ? Öyleyse ona zaman ayır buna zaman ayır ...

EA: Yok, öyle değil tabi, yaptığı işlerle alakalı. Çünkü o çok zor bir şey, hakkaten işi kendin yapmıyorsan, bir teknolojiyi uzaktan takip ediyorsun, ne olup bittiğini biliyorsun, o konuda yazacaksın ama o işi yapmıyor olacaksın. Bu çok zor. Blog'da paylaştığın şeyler gerçekte karşılaştığın problemler, onları nasıl çözdüğün, best practice'leri nereden öğrendiğin, ne yapılması gerektiği gibi konulardan oluşuyor. Bir örnek proje sunuyorsa kendi blogunda, bir şey anlatıyorsa bil ki onu müşterisine yapmıştır, kodları orada gösteriyordur. Kimse blog yazısı yazmak için kod yazmaz yani.

HE: Github da böyle, SO da böyle, bunlar içerik üretenlerine para dağıtmıyorlar. Kim, nasıl, neden bir şey üretiyor, katkıda bulunuyor ? Bu iş değil mi ya ne kadar seversen sev?

EA: Biraz ego tatmini var bunun içinde. Kendini gerçekleştirdiğini hissediyor yazılımcı. Benim bugüne kadar katkıda bulunduğum proje sayısı çok azdır. İşten vakit bulamıyorum, bir sürü işim var, github'daki ya da başka yerlerdeki projelere. Forklarım, orada durur, kullanırım. Üzerine bir değişiklik yaptıysam, private repo'ma atarım. Onu da milletten gizlemek için değil, kodumu görmesinler diye [gülüşmeler]. hani nasıl bir leş yazmış demesinler diye. Katkıda bulunduğun zaman çok dikkatli yazarsın, her şeyine dikkat

edersin. Dikkatli yazmak derken illaki best practice'den bahsetmiyorum, coding style'dan bahsediyorum. İşte indentleri 4 space içeride mi yazdın, fonksiyonun içerisindekileri tek tırnak mı çift tırnak mı kullandın, string i tanımlarken virgülden sonra boşluk mu bıraktın. Ben mesela bunlara çok dikkat ederim, işe alımlarda ben de giriyorum mülakatlara, kodunu inceliyoruz, direk eliyoruz, virgülden sonra boşluk bırakmamışsa, indentlere dikkat etmemişse, direk elerim. Çünkü bu kültüre aykırı, pis iş yapıyorsun sen, senden sonra gelecek olanı düşünmüyorsun. Çünkü o kodu sen yazacaksın işe girdiğin zaman, yarın bir gün çekip gideceksin, bizim başımıza kalacak. Koda baktığımda kargacık burgacık bir şeyler görücem. O yüzden 80 kolonu geçmemelisin, çünkü ben o kodu A4 kağıdına çıktı alıcam, taşmamalı o kod. Bunların hepsine dikkat etmelisin. Şimdi fork etmişsin bir projeyi, senin ihtiyacını görmemiş, patch etmişsin içindeki kodlarını değiştirmişsin, yorum satırlarını yazıp koymuşsun. Ama o hızda ben her zaman dikkat ederim böyle şeylere de, best practice' e dikkat etmemişsin, loopların içinde sql'leri yapıvermişsin, müşteri bekliyor çünkü. Bugün yapıyım, sonra düzeltiyim diye. Onu nasıl pull request yapıcaksın sonra, öyle olunca private repo'na atıyorsun.

HE: Biraz daha zamanın olsa ?

EA: Tabi ki, yapıp gönderirim. Çünkü her şey ihtiyaç'tan gelişiyor. Kimse şunu demiyor, ben de böyle bişi yapıveriyorum demiyor. Bugün bu ihtiyaç oluyor, onu yapıyorlar. Hiç bir proje son gün olduğu haliyle tamamlanmıyor. Ego tatmininin yanında o da var. İhtiyacın oluyor, kendin için yapıyorsun, sonrasında pull request'te bulunuyorsun. Bakın böyle bir şeye ihtiyacım oldu, yaptım, bakın diyorsun. Bazen kabul etmeyedebilirler, derler ki bu çok kosmopolit bir ihtiyaç değil, çok özel, sen al bunu kendi forkuna push et, bize karıştırma da diyebilirler. çünkü jenerik olmaya da dikkat ediyorlar o konuda, yazarken de jenerik olmaya özen gösterirler. Bir yandan da para durumu var. Dedim ya her satır kod para diye. Vakit, vakit zaten direk para.

HE: Kimin vakti ?

EA: Yazılımcının vakti, işverenin de vakti, onun da müşterisi bekliyor çünkü. Onu tamamlamadan başka işler de geliyor. Onları da planlaması lazım, o işin bir önce bitmesi lazım. Senin öğrenme vaktin, kodu yazma vaktin, onu deplore etme vaktin, bunların hepsinin çok kısa olması gerekiyor.

HE: Süreci şekillendirenler esasen bunlar mı yani ?

EA: Evet, her şeyi ihtiyaçlar, müşterinin istekleri ve para belirliyor.

HE: SO'dan konuşuyorduk, sen kendin kullanıyor musun ?

EA: Tabi ki, SO olmadan kod geliştirilmez ki.

HE: SO olmadan önce nasıl kod geliştiriliyormuş peki ?

EA: Çok zormuş abi. Şöyle, mesela benim çalıştığım şirketteki müdür o yıllardan gelme. 17 yıldır bu sektörde. Google'ın iş teklifi yaptığı bir adam, öyle birisi. Bir link'e href vermek için arayıp bulman gerekiyordu bilen adamları, veya koskoca kitapları karıştırman gerekiyordu diyor. Öyleymiş. Ben ilk geldiğimde çok mızırdanıyordum, ya bu kod böyle yazılır mı ya diye. 2004'de yazılmış bir kod, mızırdanıp duruyorum falan. O kod SO yokken yazıldı demişti bana. [gülüyor] Öyle bir durum yani.

HE: Sektördeki üretime katkısı baya fazla yani ?

EA: Tabiki, müthiş katkısı var. SO şeydir, Google'dan daha fazla önemlidir yazılımcılar için. O olmadan olmaz. Çok fazla SO kadar büyüğü yok, quora falan var ama...

HE: Bu sitelerde daha çok web ve mobil üzerine bilgi alışverişi oluyor heralde, değil mi ?

EA: Evet, ama sektör de daha çok web ve mobil üzerine gidiyor. Kimin ihtiyacı çoksa, bunu da ihtiyaç belirliyor. Soru soranlar da, yanıtlayanlar da bu alanlarda çalışanlar oluyor. Çünkü desktop application teknolojileri çok hızlı gelişmiyor, daha yavaş gelişiyor. Major değişiklikler 5 senede bir oluyor. Web'de öyle değil, sürekli sürekli... Açık kaynak teknolojilerin çoğu web alanında, Linux olduğu için. Diğerleri desktop olanlar, hep kapalı kaynak, geliştiricileri geliştirdikleri sürece gelişiyor.

HE: Peki sektörü ne döndürüyor, kim size iş veriyor, neden veriyor?

EA: Web sektörü için konuşursak en büyük neden, bu işin dinamosu reklam, duyulmak. Biz google seo işlemleri de yapıyoruz, lokal seo yapıyoruz. Search engine optimization. Mesela birisi Los Angeles'ta arıyorsa birisi, Los Angeles'taki firmayı, bizim sitesini yaptığımız adamı bulması gerekiyor. Bunun için Google'ın bazı kriterleri var, onları bilmek gerekiyor. En büyük dinamosu dediğim gibi reklam, duyulabilirlik, milletin sana ulaşabilmesi, yani elindeki ürünü satma ihtiyacı bu, aslında en temel ihtiyaç bu. Pazarda malını satarken bağırman gibi bir şey bu. Web sitesi senin dışarıya bakan yüzün, milletin sana ulaşması, hangi işi yaptığını anlaması, efektik olarak ulaşabilmesi. Bizde öyle işlemler var ki, adam göz ameliyatına girip girmemesi gerektiğini bir form üzerinden öğrenebiliyor artık, bizim geliştirdiğimiz bir form sistemi ile. Giriyor işte, daha önce şu ilaçları kullandım, uzağı göremiyorum, yakını göremiyorum, şu lensi kullanıyorum falan, sonunda diyor ki katarakt ameliyatı için uygunsun, veya göz çizdirme ameliyatı için uygunsun.

HE: Hastane ya da muayenehanedeki hizmetlerin bir kısmı da internetten alabiliyorsun yani ?

EA: Evet, bir kısmı sağlanabiliyor. Büyük şeyi reklam. Reklam sektörü de artık, yazılım sektörü gibi web hizmetleri vermeye başladılar. İşin grafik kısmıyla başlayıp, yazılım kısmına gelince outsource ediyorlar yazılım firmalarına. Artık neredeyse bütün reklam firmaları da bu hizmeti sunuyorlar.

HE: Sizin şirkette dizayn yapan insanlar var mı ?

EA: Var evet. Front end grafikerler de var, front end takımı da var.

HE: Kaç kişi var şirketinizde çalışan ?

EA: Biz burada 20 kişiyiz abi.

HE: Kaçı yazılımcı, kaçı grafiker ?

EA: Biz 6 kişiydik, 3 kişi kaldık. Şu an zor durumdayız, arayıştayız. Eğer varsa görüştüğün yazılımcılardan boşta olan ...

HE: Niye 3 kişi kaldınız peki ?

EA: Bizim çalışma saatlerimiz Amerika ile senkron çalıştığımız için 12-9, öğlen başlayıp akşam bitiriyoruz. Evlenenler oldu, onlar için uygun olmadı, ayrıldılar. Bir tanesi mesleki tatminsizlikten ayrıldı. Ben daha fazla PHP yazamayacağım dedi, gitti. O gittikten sonra biz Python'a başladık. Sağlık problemleri vardı. Şu an 3 kişi kaldık.

HE: Küçülme değil yani, hala 3 kişi arıyorsunuz ?

EA: Değil, değil. Tersine çok işimiz var.

HE: Geri kalanlar ne iş yapıyor peki ?

EA: Proje yöneticimiz var. Amerika ile iletişimi sağlayan, kanada doğumlu bir çocuk. Türk ama orada doğmuş büyümüş. İngilizcesinin native olması gerekiyor. Editörler var, içerik editörleri. Front end takımı var. Grafikerler var. Öyle...

HE: Yazılımcı, onun üstünde takım lideri gibi bir organizasyon var mı ?

EA: 3 kişi olduğumuz için bir tane takım liderimiz var, takım yok çünkü. Onun üstünde direkt patron var. Patronu şey gibi düşünebilirsin, CTO gibi düşünebilirsin. Kendisi bizden daha fazla çalışan birisi. ODTÜ finans mezunu, sonra Harvard MBA yapmış. Orada girişmiş bu işlere.

HE: Kod'dan anlıyor mu ?

EA: Yaptığımız her şeyi kod be kod bilmiyor, biraz PHP biliyor, hosting'i çok iyi biliyor. Sunucu, sistem adminliği falan yapmış. O konuda biraz rahatız, ama biraz da rahat değiliz. Çünkü ne yapıp ne yapamayacağımızı biliyor. Bu teknoloji daha hazır değil yapamayız diyemiyoruz. Dökümantasyonu önümüze atıp gidiveriyor, yutturamıyoruz yani. Öyle bir durum var ...

HE: 6 kişi bu ölçekte bir iş hacmi için az değil mi ? Böyle olmasının sebebi ürünün bitmiş olması mı mesela ? Ürün ilk anahatlarıyla yapıldığında sen var mıydın, o zamanda 6 kişi miydiniz ?

EA: Evet. Bazı müşterilerimize fabrikasyon ürün hizmeti veriyoruz, üzerinde ufak tefek değişiklikler yapabiliyoruz. O ürünün de yenisini geliştirdik, o zaman ben de vardım, 6 kişiydik. Ama bunun yanında ana ürünün dışında, herkesin kendisine ait projesi de var. Benim bir tane var, müdürün var, içerideki bir arkadaşın 2 tane var. Ama o seri üretimdeki ürüne hiç karışmıyor falan. Aslında ihtiyacımız çok var. Ama sektörde adam yok.

HE: Gerçekten öyle mi ? Az önce PHP yazan insanların bolluğundan bahsetmiştin ?

EA: İşte var ama onlar ihtiyacını karşılamıyor. Genelde öğrenmeyi bilmeyen insanlar oluyorlar. Çünkü senin teknolojin sürekli gelişiyor, her an dökümantasyon okumalısın. Ofise gelip telefonuyla oynayan adam sana yaramıyor. Biraz okuyan adam olmalı, biraz matematik kafası olmalı. Bizde kimse bilgisayar mühendisi değil. Ben kimya mühendisiyim, müdürümüz makine mühendisliği terk. Öyle bir durum var. Front endcilerimiz, grafikerimiz siyasal bilimlerden terk.

HE: Ben de geliyim demiştim şakasını yapmıştım ama sizin orası gerçekten de uygunmuş buna.

EA: Front end, jquery, javascript yazan bir arkadaş, benim üniversiteden arkadaş, kimya mühendisi.

HE: Nasıl oldu peki, nasıl işe girmişler ?

EA: Benden sonra geldiler onlar, benle birlikte girdiler.

HE: Nasıl alındılar ?

EA: Müdür aldı, referans oldum aldılar.

HE: Seni nasıl aldılar peki?

EA: Beni alırken mülakata girdik, bir kaç soru sordular, tamam gel başla dediler. Ama ben çocukluktan beridir uğraşıyordum. Bir kaç senedir uğraştığım bir şey değildi. Orta okuldayken kendi yazdığım IRC script'im vardı, kendi chat odam vardı. O dönemler oydu. MSN yokken chat odaları vardı, benim de odam vardı.

HE: Bilgi yarışması falan yapıyor muydun ?

EA: [gülüşmeler] evet, soru cevaplar ...

HE: O ilgi alaka nasıl gelişti ki ? Bende vardı, bir çok insanda vardı bilgisayar ama oyun oynuyordu insanlar, herkes kod öğrenmeyi merak etmedi ?

EA: Herkes yendi beni. Çok fena sıkılmaya başladım oyunlardan. Öyle olunca daha sonra bakmaya başladım, hazır kodlar kullandım. Yavaş yavaş okumaya başladım. Oyunlardan öğrendiğim ingilizceden ingilizce dökümantasyonlar okumaya başladım. Oyundan öğrendiğin ingilizceyle bir aşinalık kurdum. Onun çok büyük bir etkisi var, internette vakit geçiren bir çocuk üzerinde. Öyle gelişti, sonrasında forumun birisinde PHP derslerini okudum. Bir tane algoritma geliştirme kitabı aldım, lisedeyken hiç bir şey anlamadım. Üniversitedeyken anlamaya başladım. Çünkü elektronik devre anlatıyor, onun analizini çözümünü anlatıyor. Ya da fizik problemleri falan... Anlamaya başlayınca heyecanlı gelmeye başladı.

HE: Daha bütünlüklü bir bilgi haline geliyor öyle olunca sanırım, devreden başlayıp yazılıma yönelince...

EA: Tabi, tabi. Mevzunun temelini anlıyorsun. Diyorum ya bir sürü PHP'ci var diye, fundamental bilmeyen adamlar. O yüzden problem oluyor. Bakmış örneklerine, ona bakarak yapmış. O adama özel bir şeyle gelince donup kalıyor, halbuki bilmesi gerek, senin bilmen gerekiyor onu.

HE: İnternet'in kendisinden kötü bir etki olarak bahsedemez miyiz, en azından bu örnekte ?

EA: Müthiş kötü bir etki, herkes bir şeyler yazıyor, best practice değil ve ders olarak yazmış blog'una. Başlayacak olanlara kesinlikle tavsiye etmiyorum, özellike türkçe blogları.

Çok nadirdir düzgün yazan. Kesinlikle okumasınlar, yabancı blogları takip etsinler. İngilizceleri yoksa, ilk önce ingilizce öğrensinler. Bu işi hemen bırakıp, şu anda klavyeden ellerini çekip, ODTÜ yayınlarının falan ingilizce yayınları var ya onlara baksınlar. İngilizce olmadan kesinlikle mümkün değil. Facebook'ta soruyolar, programlama yapmaya hangi dille başlıyım diye soranlar için cevaplıyorum, cevap ingilizce, ingilizce ile başlayın. Kaynak yok, Türkçe kaynak yok. Bir kaç blog'dan öğrendiklerinle kalırsın. Halbuki senin pdfleri satın alıp, best practiceleri, programcılık kafasını öğrenmen lazım, algoritma problem çözme yetini geliştirmen gerekiyor. Çok fazla algoritmik problemlere bakman gerekiyor. Sunucuyu öğrenirsin, Linux'u öğrenirsin, PHP'yi nasıl kurucağını öğrenirsin.

HE: Algoritma öğrenmen gerekiyor mu gerçekten, demek istediğim sektörde o kadar ihtiyaç oluyor mu algoritma yazmaya etmeye ?

EA: Kesinlikle gerekiyor. Her zaman sıradan işler gelmiyor çünkü. Bir tane projemiz var mesela, SO gibi düşün. Benimle aynı soruyu soran adamları göster diyor mesela, ya da aynı soruyu soranlar içerisinde aynı cevapları alanlar, eşleşenleri bul diyor. Göz hastaları, aynı sorunları yaşayanları bul diyor. Müşteri gelecek siteye, soru soracak, bir tane cevap alacak, aynı cevabı başkası da almışsa, o profilleri gösterecek mesela. Bunun gibi şeylerde matematik gerekiyor, algoritma bilmeden nasıl olacak. Mümkün değil. Bİr kere mevzuyu soyut düşünmen gerekiyor. Ben her zaman söylüyorum da dalga geçiyorlar benle. Aristo'nun kategorisini bilmen gerekiyor, kategoriyi çok iyi bilmen gerekiyor. Bir class sen tanımlamadan önce yok. Sen tanımladığın anda varoluyor o. Kod var, class'ın kodu var, fakat sen onu çağırmadan, tanımlamadan, bir instance'ını örneğini oluşturmadan önce yok. Bunun ayrımını yapabilmen lazım. Array datayla String data'nın ayrımını yapman gerekiyor. Bunları anlamada mühendisliğin etkisi büyük. Sen de biliyorsundur, müthiş bir etkisi var bunun. Mat1, Mat2, Mat3, Mat4 almışsın, reaksiyon dinamiği almışsın, sen statik almışsın, onlarca benzer problemlerle uğraşmışsın. Bu sende muhteşem bir gelişim sağlıyor. Boşuna okumuşsun diyorlar, kesinlikle boşu boşuna okumadım. Öyle bir şey yok. Mühendislik eğitim olan kesinlikle bu işi yapabilir.

HE: Konuştuklarımız içerisinde bir şey dikkatimi çekti. Sektördeki insanların kendilerini geliştirmeleri gerektiğinden bahsettin, bu ne zamana kadar böyle sürecek, sürmeli ? Emekli olana kadar mı ?

EA: Emeklilik yaşı 65 ama ben 45 yaşından sonra kendimin yazılımcı olarak çalışabileceğini düşünemiyorum. Dikkatin dağılacak, eskiyeceksin, entropi var yani evrende [gülüyor].

HE: Nolacak peki ?

EA: Eğer, yönetici olarak çalışmaya devam etmediysen, ya da kendi işini kurmadıysan işsiz kalacaksın. Bu yani. Opsiyonlar bunlar ya kendin iş kuracaksın ya da yönetici olarak devam edebileceksin.

HE: Bunların şartları ne peki ? Nasıl yönetici oluyorsun, nasıl kendi şirketini kuruyorsun ?

EA: Şöyle paran varsa kendi şirketini kurabiliyorsun yoksa kesinlikle kuramazsın.

HE: Niye öyle ki ? Diğer sektörlere kıyasla bu sektördeki yatırım sermayesi daha düşük değil mi ?

EA: Tamam milyon dolarlık makinalar almıyorsun ama 5 tane işçi çalıştıracaksın, 5 tane monitor alacak olsan 3000 lira tutar. Bende yok yani o para. [gülüşmeler] O da sadece monitor.

HE: Peki bu teknogirişimcilik oldukça popüler, kim nasıl girişiyor bu işlere ?

EA: Hiç bilmiyorum. Bunu yapmak için biraz sıkışmak gerekiyor sanırım. Benim gibi rahatı yerinde olan adamlar... Parasız kalacaksın, sıkışacaksın, aç kalacaksın, o zaman kanırttırırsın, zaten açsın, bi kaç ay daha yaşar o işi kurarsın. Rahatlık insana iş kurdurtmuyor.

HE: Yönetici nasıl olunuyor ?

EA: Yöneticilik, teknik danışmanlık. Bunun için bir kere mevzuyu çok iyi bilmek gerekiyor. Oturup kod yazmıyorsun belki ama teknolojiyi çok iyi tanıman, yenilikleri bilmen, riskleri iyi analiz etmen gerekiyor. Hakkaten büyük işler büyük paralar kaybettiriyor. Milyon dolarlara satılıyor yaptığın iş, e ticaret sitesi, yapmışsın satılıyor. Senin yazdığın kod'dan, kullandığın teknolojiden kaynaklı en ufak hata her şeyi mahvedebilir. Aslında riskli bir iş ama yaşlılar için uygun. Hem tecrübe gerektiren bir şey. Neyi nasıl yapmaman gerektiği, nasıl yapman gerektiğinden daha önemli bu sektörde. O da zamanla öğrenilen bir şey. Yaşlılar için en uygun olan iş bu. Yoksa 45 yaşında klavye başında olmak sıkıntı, gözlerin bile görmeyebilir.

HE: Var mı o yaşta kod geliştiren ?

EA: Var heralde, görüyorum ama 45 yaşında mı onu da bilmiyorum. Yaşlı başlı adam, 50 yaşında bile olabilir. Vardır ama çok azdır. Bu şundan da kaynaklı, bu sektör yeni yeni canlanıyor. Henüz görmedik öyle yaşlıları. Bir kaç tane var, bundan sonra görücez artık. O yaşlarda kod geliştiren belki Amerika'da falan vardır, Türkiye'de çok yok. Angular js. yazan bir ihtiyar bilmiyorum, düşünemiyorum. [gülüyor]

HE: Maaşların da yaşa göre artması gerekiyor sanırım. Bu işler nasıl gelişiyor sektörün genelinde ?

EA: Hiç bilmiyorum açıkçası. Ben çok fazla işverenle tanışmadım. İlk defa buraya girdim. Mezun oldum, öncesinde freelance işlerim vardı ama mezun oldum buraya girdim. 1buçuk sene falan oldu. Burada yeniyim. Ama maaşım iyi, bana göre en azından. Tabi ki daha iyi maaşlar vardır. Ben 3200 lira alıyorum. Müdürümün 10binden fazla aldığını biliyorum ama. Öyle bir durum.

HE: Bu ilk hali mi, zam aldın mı, şirkette nasıl işliyor maaş artışları ?

EA: Bundan öncesinde de çok düşük değildi. 2700 alıyordum ilk girdiğimde de.

HE: Piyasaya göre daha iyi bir maaşla başlamışsın sanırım.

EA: Ya şöyle, eğer hakkaten iyi bir yazılımcıysan bu sektörde para var. Düşük para alıyorsan ama buna rağmen iyi yazılımcıysan yanlış bir şirket seçimi yapmışsındır ya da neler yapabileceğini bilmiyorsundur, sektörü iyi tanımıyorsundur.

HE: Senin planlarda ne var peki ?

EA: Önümde askerlik var. Onu aşmam lazım, şu an kendi şirketimi kurma eğilimdeyim. Onu aştıktan sonra da kendi şirketimi, e-ticaret sitesi kurma eğilimindeyim.

HE: Onu nasıl geliştiriyorsun ?

EA: Bir taraftan dışarıda kendi müşterilerime iş yapıp, oradan kazandıklarımla tahtakaleden mal alıyorum. E-ticaret sitesini geliştirmek bişi değil. Onu yaptım, bitti. Ürünü girip, siteyi tanıtıp, satışı yapması kaldı.

HE: O zaman yazılımın dışına çıkacaksın ?

EA: Evet, ticarete atılacağım, artık keyfi olarak yazılım geliştirmeye devam edeceğim.

HE: Seviyor musun yazılım geliştirmesini gerçekten, napıyorsun mesela ?

EA: Oo, çok seviyorum. Githublardaki repoları gezerim, forklarım. Katkıda bulunurum, öyle takılırım.

HE: İlla bir amaca hizmet etmesi gerekmiyor yani ?

EA: Elbette müşteri çıktığı müddetçe web işi yapmak isterim. Biraz özel işler olmasını istiyorum tabi. Kurumsal web sitesi denilen şey artık yapılmıyor bile, hazır kendi oluyor. Planlarım böyle de belli olmaz, bakmışsın 15 sene sonra da yazılım geliştiriyor olabilirim. Bizden daha önce ayrılanların gittiği şirketler de cezbediyor. Sony'e gitti bir arkadaş. Dürtüyor onlar da biraz, ayrıl diye. Ama ben çalışma ortamını seviyorum, ofisimiz tatlı bir yer, az kişiyiz, aile gibiyiz. İşveren çalışan mantığı yok bizde. Çok şükür öyle değil.

HE: Çalışma saatleri nasıl, biraz bahsettin ama açarsak, molaları falan nasıl ayarlıyorsunuz ?

EA: Çalışma saatleri 12-9 zaten. Bize her zaman ara zaten. Home ofis, yemeği de orada yiyoruz. Mutfağımız var. Ara diye bir şey yok bizde, şu zaman ara diye bir şey yok. Yemeğini ye, dışarı çık gel sorun değil, deadline'ı yetiştir yeter.

HE: Sıkıştığınız oluyor mu ?

EA: Sıkıştığımız oluyor, haftasonu çalıştığımız oluyor. Çalışmasak oluyor ama kendimizi rahatlatmak için çalışıyoruz. Sürede sıkıntı yok ama işe gitme süresinde sıkıntı yok. İlla 12'de orada olacaksın diye bir şey yok, ben 1-1buçuk gibi gidiyorum ama 9'da çıkıyorum. [gülüyor] Ama evden devam ediyorum, bazen hiç gitmiyorum evden çalışıyorum, bazı günler.

HE: Meseleye biraz iş bitsin de ne şekilde biterse tarafından bakıyorlar yani, öyle mi ?

EA: Tabi, tabi. Ofiste işin yoksa hiç gelme bile. Öyle gelmeyen bir arkadaş var. Bir tanesi skype'dan yazıyor, şimdi geliyorum diye. Patron dur gelme diyor, bir iş gönderecem sana orada halletmen buraya gelmenden daha şey... bir arkadaşımız vardı, adadan ev tutmuştu, oradan çalışıyordu. Sürekli adadaydı, bu mesleki tatminsizlikten ayrılan sadece cuma günleri geliyordu. Bir tanesi projesi vardı, Amerika'dan, sürekli onla uğraşıyor. Proje yönetimini yaptığı yerden müşteriyle iletişimini kendisi kuruyordu. Herhangi bir proje yöneticisiyle alakası yok. O yüzden gelmiyordu. Ben her gün gidiyorum, çünkü front end takımıyla da iletişim kuruyorum. Benim durumum biraz fullstack, backhand yapıyorum ama front end'de yapıyorum, javascript yazabilen bir python geliştiricisiyim. PHP çok iyi biliyorum. Python konusunda baya ilerledim.

HE: Yaptığın işlerde iyisin yani. Piyasada konumun ne ? Senin yerine başkasını bulabilirler mi ?

EA: Bulabilmeleri zor biraz. Bulabilseler yanıma alacaklar zaten.

HE: İlginç, sen mutfaktan yetişmişsin bir de.

EA: Aslında mutfaktan yetişen daha iyi. Biz bilgisayar mühendisi tercih etmiyoruz zaten.

HE: Senin nasıl buldular, nasıl güvendiler bilgine, becerilerine ?

EA: Şöyle, ben başvurdum bir ilan sitesinden. Çağırdılar, gittim. Bir ay sonra başlayacağımı söyledim. Tamam dediler. Başladım yani.

HE: Kendini sonrasında gösterdin o zaman ?

EA: Tabi tabi. Zaten belli oluyor onlar. Yazılımcıyla oturup konuştuğunda belli oluyor onlar. Belli sorular var, onlardan alacağın cevaplar yeterli oluyor. Güvenilir olması sana kalmış, biraz insan tanıyorsan, oturmasından kalkmasından belli oluyor onlar da.
...

Çok basit sorular koyduk internete, ilan sitelerine. 2 tane de soru sorduk. Çok basit sorular. Bir tanesi CSS ile diğeri de Jquery ile ilgiliydi. Ama basit sorulardı. Kimse ilanı görüntülememiş soru var diye. Soruları bi kaldırdık, 150 kişi görüntülemiş. 150 tane marjini bilmeyen, CSS'i bilmeyen jquery i bilmeyen yazılımcı varmış. Bunlar da senin işine yaramaz.

HE: Firmanın ürünleri ne kadara sattığını biliyor musun ?

EA: Hiç bilmiyorum abi.

HE: Konuşulmuyor mu bunlar ?

EA: Duyumlar alıyorum ama ... Bir tane blog sitesinin 1500 dolar olduğunu biliyorum, kurumsal web sitesinin 5000 dolar olduğunu biliyorum. Katarakt ameliyatı için olan formun siteye eklenmesi 400 dolar. Ama Amerika müşterisi bunlar tabi.

HE: Biraz daha fabrikasyon olduğu için sürümden kazanmak söz konusu sanırım ? Bakım işleri de sizde mi ?

EA: Evet, aynen, bizde öyle. Otomatize ettik her şeyi, her siteyi tek bir kanala bağladık. Önceden yataydı, her müşteriye bir tane site, bir kurulum yapıyorduk. Şimdi mevzuyu dikey hale getirdik, tek servisten data çekiliyor. Problem orada çıkarsa çıkıyor, oradan düzeltince hepsinde düzeliyor. Yalnız bu sefer de oradaki sunucuyu sürekli ayakta tutmak gerekiyor. Orası giderse sitelerin hepsi gidiyor. Onu yapmamız gerekti, büyük risk aldık

ama yapmamız gerekiyordu. PHP'den Python'a geçtik hem de problemleri dikey hale getirdik. 10 numara 5 yıldız oldu artık.

HE: İşle ilgili ne düşündüğünü az çok konuştuk ama açabiliriz daha da, iş yaratıcı mı , teknik olarak zorlayıcı mı , rutin mi ?

EA: Çok değişiyor. Yer yer sıkıcı, yer yer yaratıcı, yer yer eğlenceli. Çok değişkenlik gösteriyor, gelen işe, gelen projeye göre. Kiminde heyecanlanıyorsun, kiminde bunla kim uğraşacak diyorsun. Çünkü kod ameleliği. Ne yapacağını biliyorsun, sadece yazman gerekiyor. Bu çok sıkıntılı bir iş. Bildiğin kodu yazman sıkıcı bir iş. Bilmediğin, ekstrem bir şey gelecek ki yeni bir şeyler katacak, seni motive edecek. Bazen öyle şeyler oluyor, bir işin nasıl yapılacağını öğrenmen gerekiyor, öğreniyorsun ama öyle kalıyor.

HE: Sadece öğreniyorsun. Sadece bilme meselesi yani ?

EA: Ego'nun büyük payı var aslında, açık konuşuyum. Her insandaki öğrenme isteğinin, merak duygusunu bastırmanın, ego kısmı, öğrendiklerini başkasına gösterirken ortaya çıkıyor da, insandaki merak duygusunu bastırma çok ön planda. Bunun üzerine bilmiyorum çalışan olur mu da, psikolojik test analiz ... Merak duygusunu bastırma, hani, onun nasıl olduğunu biliyorum. İnsanı dürten oradaki gizem. Yazılım bilmeyen insanların bilgisayara nasıl baktıklarını gözlemledim. Sanki çok gizemli şeyler oluyormuş gibi duruyor ya bilgisayar. Çok büyük işler oluyor. Onun nasıl olduğunu öğreniyorsun ya, sanki bütün sistemi çözmüş gibi hissediyorsun.

HE: Pekala büyük işler geliyor mu ?

EA: Yoo [gülüyor]. Sürekli yazdığın için sıradanlaşıyor, başkaları için büyük işler belki de senin için sıkıcı hale gelmeye başlıyor. Artık daha derine inmek istiyorsun.

HE: Sıfırdan birisini aldınız diyelim, herkesin öğrenebileceği bir şey mi yazılım geliştirirken yapılanlar ?

EA: Biraz matematik kafası, sıralı düşünme olmalı. En önemlisi bu. Çünkü algoritma kurmak, böyle bir kafa yapısı gerektiriyor, şu olacak bu olacak, bu olmazsa bu olacak gibi düşünmen lazım. Bunları hemen görebilmen lazım. Refleks haline gelmesi lazım bu düşünce sisteminin. Bir şeyin hangi problemlere sebep olacağını baştan hissedebiliyo olman lazım. Tabi ki küçük şeylerde böyle. Büyük ölçekte oturup şemasını çıkarıp test ederek görmen gerekiyor. Pizza yapan adamın hamura nasıl şekil verdiğini bir yerden sonra düşünmemesi gibi. Özel bir pizza yaparken düşünmeye başlar ama ara işlemlerin hiç birisini düşünmez. Böyle olması lazım biraz. Soyut düşünmeyi, hayallemeyi iyi yapabilmesi lazım. Fonksiyon dediğinde onu somut bir şeye benzetebilmesi lazım.

Değişkeni de öyle. Değişken diyorsun, değişken dediğinde onu mühendis anlar, x görmüş y görmüş adam.

HE: Bir problemle karşılaştığınızda izlediğiniz standart bir süreç takip ediyor musunuz, napıyorsunuz ?

EA: İlk başta bütün debug toollarını kullanıyorum. Hatanın neyden kaynaklandığını öğrenmek için. Zaten bunu öğrenmeden sorunu başkalarına sormak işi başkasına yıkmak demektir. Öyle bir soru gelirse de yanıtlamıyorum. Diyor ki şurası çalışmıyor, nereden kaynaklandığını bir bul ondan sonra beraber bakalım. Büyük ihtimal nereden kaynaklandığını bulduğunda sorunu çözeceksin. Bul yine çözemedin gerçekten bir bilgi eksikliğidir, o zaman beraber bakabiliriz, ya da ben bilmiyorumdur o zaman, başkasına sorabilirim, SO'da arayabilirim falan filan. On the fly gelişiyor zaten, büyükbir kriz olduğunda zaten herkes kolları sıvayıp ona girişiyor. Öyle oturup detaylı araştırmaya da, tartışmaya da vakit yok aslında, farklı farklı yelpazelerde işler var. Ben aynı gün içerisinde 5 farklı projeye kod yazdığımı biliyorum.

HE: Mesainin tamamı boyunca çalışabiliyor musun ?

EA: Yok, hayır. Öyle olmuyor, çok kaytarıyorum. Yemek oluyor, Türkiye'yi kurtarıyoruz, futbol konuşuyoruz. Yemekler zaten ofiste yeniliyor. Biraz facebook, youtube, onu ona gönderiyorsun, bunu gördün mü, o sana bunu gönderiyor, gördün mü diye. Tam yoğunlaştığın arada mesai bitiyor [gülüyor].

HE: Emrah hem böyle diyorsun hem günde 5 projeye kod yazdığını söylüyorsun, ikisi bi arada nasıl olabiliyor ki ?

EA: Çalıştığın zaman hakkaten çok sıkı çalışıyorsun. Biraz da hızlı olmakla alakalı, kulaklıkları takıp, dünyayla ilişkiyi kesip, buluyorsun. Bazen sabahtan akşama kadar yazdığın da oluyor, bir gün hiç yazmadığın da oluyor. Değişiyor, değişkenlik gösteriyor. Deadline'a yetiştirebileceğini düşünüyorsan projeni, bu tarihlerin hepsi de belli proje yönetimi sisteminde, bakıyorsun benim şu kadar vakitte şunları bitirmem gerekiyor diye. Onları yetiştirebileceğini düşündükten sonra problem yok abi.

HE: Genel olarak düşünebilirsin ya da kendi yaptıklarınızdan da düşünebilirsin siz ne üretiyorsunuz, ürün nasıl kullanılıyor, kime ne fayda sağlıyor ?

EA: Yazılım dediğin şey, sattığın şey aslında bilgisayarda 5 sayfa kod , 5mb 10 mb kod. Bu nasıl satılabilir bir şey... Aslında sattığın şey bir menfaat. Bir ürün değil, sen soyutlama ile onu ürün yapıyorsun, ürün diyorsun. Bu nasıl bir menfaat peki ? Normalde bir hasta sana telefonla arayıp ulaşıyordu. Telefonun görevini bir bilgisayar ekranından hastaneye ulaşım

olarak ayarladın. Doktorlar normalde o kayıtları alıp deftere yazıyorlardı, sen o kayıtları database'e yazıp, onlara güzel analiz ederek sundun. Bu bir menfaat. Sattığın yazılım, kod bunu sağlıyor. Sayfalarca defter tutmayıp, kayıtları orada görebiliyorsun. Bu bir hasta kayıt formunun örneği. Bunun dışında mesela taraftar grubuyla alakalı bir videoyu paylaşıyorsun twitter'da, oradan gelen rating o adama para kazandırıyor. Bu bir menfaat, otomatize ediyor işleri. Sen normalde gidip tek tek videoları paylaşacaktın, şimdi birisi videoyu upload ettiğin anda herkese gönderiliyor, bütün üyelere mail gidiyor. Bunların hepsi menfaat.

HE: Bir yazılım uygulaması hangi sayede menfaat sağlıyor ?

EA: İşler hızlandırması, otomatize etmesi... Bunun dışında bir faydası var mı bilmiyorum. En büyük etkisi, zaten teknolojinin etkisi öyle, işleri hızlandırması. Bunun dışında sosyolojik yan etkileri, hani makinalaşma falan onlara hakim değil. İşleri hızlandırdığı kesin, daha derli toplu hale getiriyor.

HE: Sen peki ne düşünüyorsun, işleri hızlandırmasının etkileri neler ?

EA: Aslında sonuçları üzerine çok düşünmedim. Bana göre hızlanması gereken sektörler var, hızlanmaması gereken sektörler var. Toplumun yararına mı değil mi orasını kestirmek için daha fazla analiz yapmak, veri toplamak gerekiyor, bakmak gerek. Kapitalizmin bir tarafına çok katkıda bulunuyorsun ama doktor da hastası ile daha fazla iletişim halinde olabiliyor. Hangisi hangisine ağır basıyor, galip geliyor, bilemiyorum. Bir de bir çok insanın işine ket vurmuş da olabilirsin böyle bir sistemi yaparak, yaptığın bir yazılımla bir insanın işten çıkmasına da sebep olabilirsin. Eskiden o defter kayıtlarını tutan adam varken artık yoktur, doktor kendisi bakabiliyordur. Çünkü yazılım yapıyor artık o işi.

HE: Yazılımcıların, bilgisayar mühendislerinin örgütlülüğü hakkında fikrin var mı ?

EA: Bilgisayar mühendisleri odasının kurulmaya çalışıldığını duymuştum. Diplomasız yazılımcıların çalıştırılmaması konusunda bir çalışmaları vardı, bence kurulmamalı [gülüyor].

HE: Sektörde hak ihlalleri yaşanıyor mu ? Öyle bir durumda başvurulacak bir örgütlenmeye ihtiyaç var mı?

EA: Öyle bir ihtiyaç var mı yok mu görecek kadar süredir çalışmıyorum. Ben de öyle bir durum yaşamadım, ihtiyacım olmadı. Olursa da öyle bir yere başvurmam. Zaten dediğim gibi yelpaze çok geniş, bu gün kendime yazılımcı diyorum ama donanım tarafında yazılım geliştirenler bana yazılımcı demiyorlar, ürün kullanıcısı diyorlar atıyorum. Anlatabiliyor muyum ? Adam şey diyor, dili geliştiren biziz, sen bizim geliştirdiğimiz dili kullanıyorsun gibi kafa var. O yüzden birbirine çok uzak insanlar var sektörde. Büyük bir yelpaze. Biz de

o adamlara moron gözüyle bakıyoruz, senin yüksek seviyeli dillerde hiç bir yeteneğin yok, transistörlerle data gönderip alıyorsun kart üzerinde ama yelpaze çok geniş. Kim kime yazılımcı diyecek ? Desktop geliştirenler ayrı, framework geliştirenler var, engine geliştirenler var bizim gibi en üst katmanda müşteriye iş yapanlar var, askeri düzeyde iş yapanlar var. Çok geniş bir yelpaze, hepsinin kendisine göredir iş hayatındaki sıkıntıları. Bu inşaat mühendisleri odası gibi değil, hepsine belki ayrı ayrı oda gerekir. Tek kanaldan gelişmiyorlar, parametreleri hesap edemiyorsun, o yüzden böyle bir odanın kurulması çok da mümkün gelmiyor bana. Türkiyedeki sektör biraz şey gibi, en azından burada, din konuları gibi, ağzı olanın konuştuğu başka bir sektör pek yok. Çünkü bir otorite yok, üniversiteler işin çok gerisinde. Teknoloji'de, özellikle yazılım tarafında, gelişmeleri akademi değil sektör belirliyor. Angular js'i google geliştiriyor. PHP'i geliştiren adam herhangi bir adam, laravel'i geliştirenler, django'yu yazanlar toplanıp bunu yapan adamlar. Üniversite bu yüzden işin gerisinde. Üniversiteden mezun adam araştırma okuma çalışma yapmıyorsa sektörde tutunması zor. Bir alana eğilmesi, kendisini özelleştirmesi gerekiyor. Yeni yeni python öğretilmeye başlandı üniversitelerde. Pascal, delphi öğretilen yerler var. Kalmadı ki bu teknolojiler. Hocalar yaşlı, akademisyenler konuya çok eğilmiyorlar. Bir iş görüşmesinde IT'nin başındaki adam bana şey demişti, üniversiteyi şöyle tanımlıyorum dedi, hocaların gözlerinin olmadığını düşün, ellerinde bir ışık, öğrencilere yol gösteriyorlar bilgisayar mühendisleri için. O da alaylıydı, başarılı birisiydi mesela.

HE: Bunun bilimsel bir kısmı yok mu peki ?

EA: Var tabi, geliştiren adamlar zaten sağlam bilim altyapısına sahip insanlar zaten. Fakat bu ülkemiz için geçerli bir şey değil. Sektör taraflı diyorum ama Google'da çalışan adamlar Stanford'dan MIT'den mezun adamlar. Orada alıyorlar bunu ama sektör içerisinde geliştiriyorlar. Benim çalıştığım alan yani web teknolojileri üzerinden konuşuyorum. Tabi ki savaşın getirdiği teknoloji de büyük, telgraf teknolojisi, haberleşme... En önemli gelişmeler askeri alanda çalışanlar tarafından gerçekleştirilmiş. Biz web'de çalıştığımız için o alandaki ihtiyaçları biliyorum. [...]

HE: Ayırarak soruyorum, mesleğinle ya da işinle ilgili değiştirmek istediğin bir şey var mı?

EA: Şu anda değiştirmek istediğim bir şey yok. Değiştirirsem bundan daha kötü olabilir mi diye düşünüyorum. Belki proje yönetimi daha iyi olabilirdi. Kendi projelerimizi kendimiz yönetiyoruz çoğu zaman. Seri işler için proje yöneticisi var. Ama farklı işler geldiğinde kendi projelerimizi yürütüyoruz. O da zor oluyor. Bir taraftan müşteriyle uğraşıp, bir taraftan kod yazıyorsun. Halbuki o işlerin halledilip, sana ne yapacağının analizi hazır yapılmış halde gelmesi gerekiyordu. Sana denecek ki, şöyle bir ihtiyaç var o yapılacak. Sen müşteriyle tartışıyorsun o olurdu bu olmazdı diye, o senin işin değil ki. Proje yöneticisinin halledip önüme bu iş bu şekilde halledilecek diye gelmesi lazım. Ben hemen o işi bitirip öğrenmeye devam etmeliyim.

HE: Şirkette bu öğrenme meselesine nasıl bakılıyor, iş tanımının içerisinde mi kendini geliştirmen peki ? Çalışanlarına eğitim veriyor mu şirketler ?

EA: Tabi ki öyle bakıyor o da, kendisi veremiyor eğitim, verebilecek öyle bir kurum yok çünkü. Dese ki şirket içi bir eğitim verilecek şuraya gidiyoruz dese kim verecek ?

HE: Amerika'ya gitseniz ?

EA: Ha, olabilir ama öyle bir şey yok. Türkiye'deki besin zincirinin en yukarısındayız heralde... Altlarında da olabiliriz ama benim daha çok işime gelirdi daha basit işler yapıyım, kalan zamanımda kendi işlerimle uğraşıyım. Burada büyük bir işle uğraşıyorsun, yapacağın teknolojiyi seçiyorsun ama hiç bir fikrin yok o teknolojide. Dökümantasyonları okumakla başlıyorsun, ufak tefek denemeler yapıp sonra büyük projeye uyguluyorsun. Halbuki ben isterim ki küçük basit bir iş gelsin sonra ben ona rahat rahat bakıyım. Best practice'ı atlıyorsun öyle olduğunda.

HE: Şirkette hiç sorun yaşadın mı, patronunla ya da çalışma arkadaşlarınla ?

- Şöyle oldu, bu da kendini geliştirmekle alakalı. Atöyle yapmaya kalktık, dedik ki atölye yapacaz, git kullanmayı bilmeyen var mı ? Bu büyük bir ihtiyaç. İstemediler, bilmediğin her yeni şeyden korkuyorsun. Aslında korkmamak gerekiyor. Git'in Türkiye'de kullanıldığını bilmiyorlar, atölyeye de katılmak istemiyorlar. Şöyle işler oluyor, bir tane proje çok büyük, kesinlikle FTP üzerinden bağlanılmaması gerekiyor. Sadece git üzerinden yönetiliyor ki, kimin ne yaptığını görelim, patladığı zaman geri alabilelim. Kimin satır satır ne yazdığını takip edelim diye git'de duruyor. Front end takımındasın, git bilmiyorsun, oradaki imaj manipülasyonunu benim yapmam gerekiyor o yüzden. Bildiğin photoshop açıp resmi düzenliyip tekrar atıyorum. O işi o kadar işin arasında ben yapıyorum, sırf sen git bilmiyorsun diye. Böyle problemler oldu ama aştık. Başka da problemler olmadı. Şey var çalışanlar arasında, normalde yapmaya alıştığının haricinde iş geldiğinde sanki 2 kat çalışıyormuş gibi hissediyor. Benim işim bu, ben işe girerken böyle dendi, başka da bir şey yapmam. Aslında maaş konusu da, iş bulma konusu da böyle değişiyor. Sen şirketi ileri taşıyorsan bir şeyler oluyor. Sen patronun, müdürün, işverenin sana bir şeyler demeden bazı şeyleri öngörüp, araştırıp çalışıyorsan, öğreniyorsan oluyor. Hakkaten hiç angular js kullanmadan iş yapabilirsin bu sektörden, jquery kullanmadan, framework bilmeden iş yapabilirsin. Kimse de sana demez, neden kullanmıyorsun, çünkü müşteri çoğunlukla arkada dönen işi bilmiyor. Sen çok yeni çok özel bir teknoloji kullanıyorsun, ama adamın gördüğü bir şeye tıkladığın bir şeye erişmesi. Bunu istiyor adam, böyle işler yapabilirsin. Ama işveren için öyle olmuyor. Sen o teknolojiyle iş hızını 2 katına çıkardığın vakit, işleri kolaylaştırdığın vakit işverenin sana bakışı değişiyor. Bunların olması gerekiyor. Yoksa gidip bir şirkette oturup, senelerce de aynı şeyi yapabilirsin.

HE: Yeni iş geldiğinde bazıları normalde yaptığından 2 kat fazla zorlanıyor'un gerekçesi ne ? Yeni bir şey öğrenmenin zorluğu mu ?

EA: Evet, aynen öyle. Yeni bir şey öğrenmenin zorluğundan. Bir de çalışanla iş veren arasında şöyle bir ilişki var. Bir tanesi ne kadar az çalışacak diye bakıyor, patron hep şüpheli yaklaşıyor, ne kadar çok kaytaracak acaba diye. Diğerinin de beni aldığım maaştan daha fazla çalıştıracak diye bakıyor. Patron kaytarmayı minimuma çekmeye çalışıyor, diğeri maaş performans oranını maksimumda tutmaya çalışıyor. Aslında bu sektörün en büyük parametrelerinden birisi kendini ne kadar geliştirdiğin, maaşına da bakmayacaksın. Bir şirkete girdiğinde kendine ne kadar çok şey katacaksın, bunlara bakman gerekiyor, burası önemli. Bunu aşamamışsan bu sektörde çalışmak çok zor.

HE: Maaş veren adam verdiği paranın karşılığını almak istiyor, çalışan da aldığı maaşın karşısında efor sarfediyor, bunun ölçütü var mı, nasıl ölçülüyor ?

EA: Bunun kesin bir ölçütü yok. Az evvel maaşlardan konuştuk, bana burada bu kadar veriyorlar, başka bir firma vermeyebilir. Bu firmanın yaptığı işle de alakalı, firmanın hacmiyle de alakalı. Büyük bir reklam şirketidir, çok büyük müşterileri vardır, bir tane sosyal medya uzmanına dünyanın parasını verebilir. Ama müşterisi az olan çapı küçük olan bir firma o yazılımcıya az miktarlarda maaş da verebilir. Bir taraftan otorite de yok çünkü, bu güzel koddur, bu çirkin koddur'a dair bir otorite yok. Ne Amerika'da var ne Türkiye'de.

HE: Peki bu işveren ile patron arasındaki mesele nasıl çözülebilir ?

EA: Şöyle; patronun kafasında aslında şu var. Tek derdi para kazanmak. Kimse kimseye karşılıksız bir şey yaptırmıyor. Kazandırdıkça şirketin bir parçası haline geliyorsun çalışan olarak. Sen kazandırdıkça o seni daha fazla motive ve mutlu ediyor. Fakat bunu hissettirmen lazım, çalışanın bunu bilmesi lazım. Yazılımcının az bulunduğu bir sektörde düşün ki çalışan yazılımcı bütün işlerini ayağa kaldırıyor. Ona yaklaşımın nasıl olur, ortak bile edersin firmaya, çünkü gitmemesi gerekiyor, birisinin daha iyi bir teklifte bulunmaması gerekiyor, hisse vereceksin, ortak olucaksın, anlatabiliyor muyum. Ama aynı işi sürekli yapan, farklı iş gelince mızmızlanan birisi için de farklı şeyler düşünürsün. Gözün dışarıda olur, yeni bir işçi bulabilir miyiz diye. Şimdi işten çıkarmalar da öyle, yeni bir işçi alınır, bütün işler ona devredilir, artık ona iş verilmemeye başlanır ve kendisi bırakır artık. Öyle de oluyor. Çünkü işten çıkarma da tazminat falan veriyorsun. Değişik politikalar da var. İş vermiyorsun adama. Yavaş yavaş hissediyor artık istenmediğini. Gerçi mobbing oluyor mu bilmiyorum böyle şeyler ama bunlar da var.

HE: Boş zamanlarında ne yapıyorsun ? Diğer çalışanlar için ekrana bakmak farklı bir şey olabiliyor ama sen zaten tüm gün bilgisayar başındasın ?

EA: Bilgisayarda devam ediyorum. Şu aralar trendim mobil teknolojiler. Mobil development konularına bakıyorum. Dışarıdan aldığım işlerle uğraşıyorum. Yine televizyon açık oluyor.

HE: Birden fazla şeyin açık olması odaklanmanı zorlaştırmıyor mu ?

EA: Yok aksine, gürültü olmalı, ses olmalı. Sessiz bir yerde çalışamıyorum ben. Bir başkası, bu adada çalışan arkadaş da sesli ortamda çalışamıyordu. Ne olursa ses olmalı benim için. Müzik, insan sesi... Zifiri sessizlikte çalışamam. Çalışırken haberleri açarım bi yandan, televizyon açarım. Spikerin konuşması lazım. İzleyeceğim dizileri ise ayrıca oturup izlerim. Yoksa sadece dönecek bir şey olması yeterli.

APPENDIX E: TURKISH SUMMARY

1970'lerden günümüze, enformasyon ve iletişim teknolojilerinin toplum ve üretimin genel yapısı üzerindeki etkinliği ve önemi hatırı sayılır oranda artmış gözükmektedir. Tarihsel bir kopuşa mı yoksa devamlılığa mı karşılık geldiği tartışılmaya devam edilse de, araştırmacılar arasında, bu teknolojilerin toplumu ve genel üretim organizasyonunu dönüşüme uğrattığı konusunda anlaşmaya varılmış gibidir. Bakıldığında, enformasyon temelli aktivitelerin tarımsal ve endüstriyel sahalardaki kullanımının artmasıyla çoğunlukla hizmet temelli yeni meslek gruplarının ortaya çıkmasından söz edilebilir.

Yazılım uygulamaları, dolayısıyla yazılım geliştirme, enformasyon ve iletişim teknolojilerinin yaygınlaşmasında en kritik role sahip unsurlardan birisidir. Modern bilgisayarların insanlar için hayati bir değer kazanması, yazılım uygulamalarının gelişiminin bilgisayarların kullanım alanlarını arttırabilmesiyle mümkün olmuştur. Enformasyon temelli ürünlerin bugünün ekonomisinin karakteristiklerinden olması sebebiyle, yazılım uygulamaları bu dönemin en kritik araçlarından birisi olarak kabul görmektedir. Dolayısıyla yazılım geliştirme sürecinin odağındaki yazılım üreticileri için emek piyasasının aristokratları olduğu/olacağı temelli iddialar ortaya atılmıştır.

Egemen diskura göre, yazılım geliştirenler entellektüel ve yaratıcıdırlar, meşgul olacakları işi kendileri seçebilir ve iş sürecini kendi kendilerine yönetebilirler. Enformasyon'un toplanması, değerlendirilmesi, organizasyonu, nasıl işleneceği gibi kompleks işlemler yazılım geliştirenlerin inisiyatifleri içerisinde görevlerdir. Fakat, yazılım geliştirme sürecini oldukça olumlu bir biçimde tasvir eden bu yaklaşımlarda kapitalist üretim ilişkileri ve/veya güç ilişkileri fark edilemez ya da göz ardı edilir. Yazılım geliştirme'nin hangi üretim ilişkileri dahilinde ve kimler tarafından gerçekleştirildiği konuları kolayca gözden kaçabilmektedir.

İfade etmek gerekir ki, sermaye'nin emek üzerindeki denetimi ve üretim ilişkileri anlamında yazılım geliştirme'nin emek süreci, üretim'in diğer alanlarıyla karşılaştırıldığında kendine özgü karakteristikler taşımaktadır. Yine de, sermayenin yeniden üretimi, birikimi ve üretim üzerindeki denetimi yazılım sektöründeki denetim mekanizmaları içerisinde yer edinebilmiştir. Dolayısıyla, yazılımcıların

emekleri üzerinde belli bir oranda otonomi sahibi oldukları iddia edilebilirse de, yazılımcılar ücretli emek mensupları olarak emeklerini satmak, haliyle kapitalist üretim ilişkilerine dahil olmak zorundadırlar.

Yazılım uygulamalarının hangi üretim ilişkileri doğrultusunda nasıl bir biçimde geliştirildiği üzerine yapılan çalışmalara Türkiye özelinde rastlamak oldukça güçtür. Yapılan çalışmaların da, sektörün daha görünür ve çoğunlukla 'ışıltılı' kısımlarına odaklandıkları söylenebilir. Yazılım uygulamalarının üretim noktasında gerçekte neler yaşandığına, sosyal ilişkilere yeteri kadar ilgi gösterilmemektedir.

Bu doğrultuda, bu çalışma Türkiye'de gerçekleştirilen yazılım projelerinin hangi spesifik teknik örgütlenme içerisinde icra edildiği, işçilerin üretim üzerindeki hakimiyetlerinin bu teknik örgütlenmeden nasıl etkilendiği, iş yerindeki denetim ilişkilerinin ne tür formlar aldığı, yöneticilerin ne tür denetim stratejileri uyguladığı, yazılım geliştirenlerin bu stratejilere nasıl karşılık verdiği ve üretim ilişkilerinin iş güvencesizliği bağlamında yazılımcılarca nasıl algılandığı bu tez çalışmasının araştırma sorularını oluşturmaktadır.

Çalışma bu araştırma soruları doğrultusunda üç argüman geliştirmiştir. İlk iddiamıza göre, üretim'in teknik örgütlenişinin aralıksız dönüşümü, yazılım geliştirme pratiklerinin standardize ve otomatize edilmesine yönelik eğilimler göstermektedir. Bu durum, üretim sürecinin kontrolünün yazılım geliştirme araçlarının kullanımlarıyla bir miktar yöneticilerin eline geçmesine yol açmaktadır. Doğrudan üretici pozisyonundaki işçilerin üretim süreci üzerindeki etkinlikleri ve denetimleri azaldıkça da, sahip oldukları vasıf ve niteliklerin piyasa değeri aşınma eğilimi gösterir, dolayısıyla yazılımcılar iş koşulları anlamındaki ayrıcalıklarını yitirebilirler. İkinci olarak, yazılım geliştirilen iş yerlerinde, yöneticiler tarafından uygulanan doğrudan denetim stratejilerinin yanı sıra, yine bir denetim stratejisi olarak çalışanlara görece bir otonominin bahşedildiğini gözlemledik. Tanımlanan bu 'özgürlük' alanının sınırları, şirketler'in karlılığının arttırılması ve karlılığın sürekliliğinin güvence altına alınması doğrultusunda şekillenmekte gözükür. Dolayısıyla, işçilerin rızasını ve işbirliğini edinmeyi hedefleyen stratejiler de dahil olmak üzere, yazılım geliştirmenin emek süreci kapitalist birikim'in gerekliliklerine uygun bir yapılanma gösterir. Son olarak, esnek emek rejiminin yıkıcı etkileri, kısa vadeli istihdam ilişkileriyle birlikte yazılım çalışanlarının üzerindeki belirsiz gelecek ve iş güvencesizliği baskısını sürekli tutmaktadır. Bu sebepten, işçiler sömürüldükleri üretim ilişkilerine tâbiiyet gösterme eğilimindedirler. Emek

piyasasında rekabet güçlerini koruyabilmek ve istihdam edilebilir kalmak adına başvurdukları birey bazlı ve piyasa güdümlü stratejiler bu ilişkilerin yeniden üretimine katkı sağlar.

Bu bağlamda, çalışmamız enformasyon ve iletişim teknolojileri endüstrisi içerisinde merkezi bir noktada konumlanan yazılım geliştirmenin emek sürecini incelemiştir. Hayatın her alanında kullanılan yazılım uygulamalarının üretiminin çok fazla akademik ilgi görmemiş olması noktasından hareketle, Türkiye'de gerçekleştirilen yazılım üretimi, sektörün çalışanları ile yapılan görüşmeler ışığında aydınlatılmaya çalışılmıştır. Yazılım uygulamalarının hangi üretim ilişkileri içerisinde nasıl geliştirildiği ve bu ilişkilerin yazılım işçileri tarafından nasıl deneyimlendiği ve değerlendirildiğine dair genel bir çerçevenin sunulmasına gayret edilmiştir. 'Yeniliğin' lokomotifi, bugünün dünyasının en temel bileşeni olan yazılım uygulamalarının üretim sürecinin kapitalist üretim ilişkileri anlamında hangi açılardan farklılıklar ve hangi açılardan süreklilikler gösterdiği araştırılmıştır. Yazılım üretiminin teknik organizasyonu, iş yerindeki denetim ilişkileri, vasıfların üretim içerisindeki rolü ve geleceklerinin belirsiz olması bağlamında yazılım çalışanlarının deneyimlediği iş güvencesizliği konuları ele alınmıştır.

Ortaya konulan bu araştırma konularına açıklık getirebilmek için, kartopu örnekleme yöntemiyle ulaşılan 21 yazılımcıyla yarı-yapılandırılmış görüşmelerin gerçekleştirildiği bir saha çalışması yapılmıştır. 2014'ün Kasım ayında, örneklemimi neye göre oluşturmam gerektiğini belirlemek için internet üzerinden bir anket hazırlanmıştır. 40 sorudan oluşan bu anket vasıtasıyla, sektörde çalışan yazılımcıların demografik bilgilerinin nasıl dağılım gösterdiği ve araştırma konularımla nasıl ilişkilendiklerinin öğrenilmesi hedeflenmiştir. Fakat, yazılımcıların sık takip ettikleri bir çok blog'tan yayınlamalarını rica etmem gibi ulaştırma gayretlerim rağmen anket yaygınlık kazanamamıştır. 1 ay sürecince 14 kişinin katılımıyla sınırlı kalınca anket çalışması iptal edilmiştir. Görüşmelerimi yaş, şehir ve çalışılan şirketlerin organizasyon yapılarının dengesini gözeterek gerçekleştirdim. Görüşmecilerim'in çoğunluğuna 1. ve 2. dereceden tanıdığım insanlar ve onların tanıdıkları vasıtasıyla eriştim.

Görüşmelerimde yazılımcılarıma hazır bir soru listesinden soru yöneltmedim. Temel konu başlıkları ve bu konular için hazırladığım sorular vardı, ama bu sorulara sadece görüşmecilerim belirlediğim konuların dışına çıktığında başvurdum. Araştırmam dahilinde kaldıkları sürece görüşmeleri büyük çoğunlukla

yazılımcıların anlattıkları ve anlatmak istedikleri üzerinden devam ettirmeye çalıştım. Konuşmaların bütünlüğünün ve akışının bozulmamasına gayret gösterdim. Görüşmeci ile görüşmeyi yapan arasındaki mesafeyi olabildiğince aza indirgemeyi amaçladım. Görüşmelerimin büyük bir çoğunluğunda yazılımcılar yardımcı olmak için gayret gösterdiler. Sorduğum soruların çoğunluğuna açık yanıtlar verdiler. Sadece, o da bazı görüşmecilerin maaş ve çalıştıkları şirketlerle ilgili sorularda çekince gösterdikleri belirtilebilir.

Görüşme yaptığım 21 kişinin 5'i kadın 16'sı erkek idi. 12 kişi Ankara'da çalışıyor, kalan 9 kişi ise İstanbul'da çalışıyordu. Sektör'ün çalışanlarının büyük bir oranda 30 yaş altı olduklarını göz önüne alarak, örneklemimi bu doğrultuda oluşturdum. Görüşmecilerimden 11 tanesi 30'un altı, 10 tanesi 30 ve üstü yaşlara sahiptiler. 10 tanesi bilgisayar mühendisliğinden mezun olmuştu. Kalan görüşmecilerim farklı bölümlerden mezun olmuş fakat kendilerine meslek olarak yazılım geliştirmeyi seçmiş kimselerdi. 7 görüşmecim farklı mühendisliklerden, diğer 4 'ü ise temel bilimler ve eğitim fakültesi bölümlerinden mezundular.

Görüşmecilerimden 12 tanesi ODTÜ mezunuydu. Geri kalanların da büyük çoğunluğu İstanbul ve Ankaradaki üniversitelerden mezun insanlardı. Anadolu'daki üniveritelerden mezunlara ulaşmaya çalışsam da bunu başaramadım. Bir çok çalışma tarafından ortaya konduğu üzere, büyük şehirlerdeki 'prestij'li üniversitelerden mezun olmak beraberinde daha avantajlı olmayı da getirebilmektedir. Bu üniversitelerden mezun olanların, en azından, Anadolu'daki üniversitelerden mezun olanlara kıyasla çalışma koşullarının görece daha iyi olduğu varsayımı yapılabilir. Bu durumun çalışmanın saha bulguları üzerindeki olası etkileri göz önünde bulundurulmalıdır.

Görüşmecilerimden 3'ü kendi şirketlerinde yazılımcı olarak çalışıyordu, 1 kişi de 'freelance' olarak serbest çalıştığını söyledi. Geri kalanlar sözleşmeli olarak çalıştıklarını ifade ettiler. 8 tanesi büyük ölçekli sayılabilecek kurumsal firmalarda, 7 tanesi orta ölçekli, 5 tanesi de küçük ölçekli firmalarda çalıştıklarını belirttiler.

Devlet'te yazılımcı ya da bilgisayar mühendisi olarak çalışmanın özel sektöre göre farklılıklar içerebiliyor olması sebebiyle, örneklemimizi özel sektörlerle sınırlandırdık. Bu yüzden, görüşmecilerimin tamamı özel sektörde faaliyet gösteren firmalarda çalışanlardan seçilmiştir. Özel sektör ve kamu sektörü arasında bir yerde duran, vakıf şirketleri; Aselsan, Roketsan gibi şirketleri de çalışmaya dahil

etmedim. Bu şirketlerin Türkiye yazılım piyasasındaki avantajlı konumları çalışanlarına tanıdıkları görece güvenceli ve ayrıcalıklı çalışma koşulları sağlamalarını sağlayabiliyordu. Bu yüzden bu şirketleri de çalışma içerisine dahil etmedim.

Görüşmelere Aralık ayında başlayıp, Mayıs ayında sonlandırdım. Süreç, görüşmelerin transkripsiyonunun yapılması da dahil olmak üzere yaklaşık 6 ay sürdü. Başlangıç aşamasında sürecin bu kadar uzun sürmesini planlamıyordum. Fakat, görüşme yapmayı kabul edecek yazılımcıları bulmam ve onlarla görüşmem düşündüğüm kadar kolay olmadı. İrtibata geçtiğim herkes yardımcı olmak istese de, yoğun çalışma tempoları sebebiyle görüşmeler için onlara uygun zamanı beklemek durumunda kaldım. İrtibat içerisinde olduğum ama zaman ayarlayamadığımız için görüşemediğim yazılımcılar da oldu. Bu sebeplerden, süreç beklediğime göre daha uzun sürdü.

Görüşmeleri her bir tanesinde 7'şer kişiyle görüştüğüm 3 parti halinde gerçekleştirdim. Her parti görüşmelerin arkasından deşifreleri yaptım, bunları danışmanımla paylaştım. Aldığım tavsiyelere göre görüşmelerde daha az ya da daha fazla üstüne durmam gereken konuları tekrar düzenledim.

İlk görüşmelerimin çoğunluğu en az 2 saat sürdü, fakat bu sürenin görüşmecilerime bağlı olarak değişkenlik gösterdiği de oldu. 3 saat süren görüşmelerim olduğu gibi 1,5 saat süren görüşmelerim de oldu. Son yaptığım görüşmelerde bazı konular hakkında aldığım yanıtların tekrar etmesi sebebiyle, bazı konuları daha hızlı geçtiğim durumlar oldu. Son parti görüşmelerimin çoğunluğu bu sebepten 1-1,5 saat içerisinde sonlandı.

Görüşmelerin 8 tanesi iş yerlerinde ya da iş yerlerinin bulunduğu binaların içerisindeki kafelerde gerçekleştirildi. Bu görüşmelerin 4'ünde iş yerlerini görebilme şansı buldum, ve çalışma ortamlarını gözlemleyebildim. İş yerlerinin hemen hemen hepsi yazılımcıların çalıştıkları bölmelere göre tasarlanmış gözüküyordu. Bölmelerind, yazılımcılar genellikle kulaklıkları takılı halde bilgisayarda gerçekleştikleri işleriyle meşgul görünüyorlardı. Yazılım geliştirmeye dair ayırt edici bir farklılık gözlemleyemedim. Bu anlamda, yazılımcıların iş yerlerinin, işlersin bilgisayarlar aracılığıyla görüldüğü herhangi bir ofisten ayırt edilebileceklerini söylemek zor .

Bu anlamda sadece bir firma diğerlerinden ayrışıyordu. Bu firmada, mobilyalardan duvarlara, iş yerinin tasarımıyla ilgili yapılan tüm seçimlerin daha farklı bir bakış açısı gözetilerek yapıldığı söylenebilir. Bu iş yerinin kullanım skalasının da farklı olduğunu gözlemledim. Mesela ofiste geçirdiğim süre içerisinde, 2 farklı drone uçurulduğuna şahit oldum. 2 tane çalışanın hatırı sayılır bir süre oyun konsolunda oyun oynadıklarını gördüm. Bu firmanın startup kültürü olarak da ifade edilebilecek, yurtdışındaki yazılım firmalarında yaygın bir biçimde benimsenen yeni iş kültürün özelliklerini yansıttığı söylenebilir.

3 kişiyle videolu görüşme yaptım. Görüşmelerimiz ara sıra teknik sebeplerle kesintilere uğradı. Fakat bu kesintiler görüşmeleri genel olarak pek etkilemedi. Görüşmecilerim video konferans ya da video görüşmeye oldukça alışkın gözüküyorlardı, bu yüzden kurduğumuz iletişim, yüz yüze görüşmenin sağladığı iletişimden pek bir farklılık göstermedi.

Yazılımcılarla genelde mesai saatlerinin sonrasında görüştüm, bi kaç kurumsal şirketi kenara koyarsak, yazılımcıların uymaları gereken giyim kodlarının olmadığını, varsa da oldukça esnek olduklarını gördüm. Erkekler genellikle kanvas pantolon, üzerine tişört, kazak ya da gündelik bir gömlek giyiyorlardı. Kadınların giyimleri de gündelik olmaları anlamında benzer özellikler gösteriyordu. Yazılımcıların kendilerini rahat hissettikleri türde giyinme serbestliklerine sahip oldukları söylenebilir.

Çalışmanın ilk bölümünde yazılım geliştirmenin teknik organizasyonunu ve bu organizasyon yapısının üretim ilişkilerine olan etkilerini tartışmaya çalıştım. Bunun için yazılım geliştirme araçları içerisinde kritik rollere sahip unsurları ele aldım. Üretimin standardize edilişi, otomasyonu ve basitleş(tiril)mesi bağlamlarında, programlama dilleri, programlama paradigmaları, programlama araçları ve teknolojileri ve internet kullanımının etkileri tartışıldı.

Bu konuların çalışmada neden yer aldıklarına dair kurduğum bağlamı biraz daha açmam isabetli olacaktır, bunun için yazılım geliştirmenin tarihçesi içerisinde biraz geriye gitmekte fayda var. Modern bilgisayarların kullanımının ilk yıllarında, yazılım ve donanım arasında bugünki gibi bir ayrışmadan bahsedilemezdi. Bilgisayarların üretimi ve gösterdiği faaliyetler söz konusu olduğunda, yazılım ve donanım birlikte ele alınan unsurlardı. İkinci Dünya Savaşı döneminde kullanılan teknolojilerin yazılımını geliştirenlerin çoğunluğu elektrik mühendisiydi mesela, ve

bu işi yazılım konusunda bir eğitim almış oldukları için değil yazılıma ilgi duydukları için yapıyorlardı. Yazılım geliştirme pratikleri de genellikle bir usta çırak ilişkisi içerisinde aktarılmaktaydı.

Donanım ve yazılım alanındaki teknik gelişmeler ile birlikte yazılımın bilgisayarları nasıl dönüştürebileceğinin farkına varılmaya başlandı. Yazılım sayesinde bilgisayarların çeşitli ihtiyaçlara göre programlanabilmesi, bilgisayarların fonksiyon gösterebileceği kullanım alanlarını arttırabilirdi. Nitekim öyle de oldu, yazılım alanındaki gelişmelerle birlikte bilgisayarların gerçekleştirebileceği işlemler çeşitlilik göstermeye başladı. Yazılım uygulamaları ve yazılımcılar böylelikle 'bilgisayar devrimi'nin taşıyıcı aktörlerinden oldular.

Bu durum bilgisayarlara girilecek direktifleri organize edecek ve bu direktifleri makinenin anlayabileceği bir programlama dilinde ifade edebilecek bir meslek grubunun, uzmanlığın varlığını gerektirmekteydi. Üzerine yapılan çalışmaların artmasıyla birlikte, yazılım akademide de ayrı bir disiplin olarak kendisine yer buldu.

Yazılım ayrı bir uzmanlık alanı olarak formelleşse de, üretim organizasyonunun konvansiyonel bir biçim kazanması hatırı sayılır bir süre alacaktı. Mesela bugün yaygın olarak kullanılan üretim araçlarına temel oluşturacak fikirler ancak 70'lerin ortasından itibaren tartışılmaya ve uygulamaya konulmaya başlandı.

Bütün bunlarla ilişkili olarak şu söylenebilir, ilk dönemlerinde yazılım geliştirme çok fazla sayıda insanın gerçekleştirebileceği bir aktivite olmaktan uzaktı. Ciddi seviyede matematik bilmek, kullanılan programlama dillerinin yetkinliklerine ve sınırlarına hakim olmak, yazılım uygulamasını karmaşıklığı içerisinde organize edebilmek yazılımcıların sahip olması gereken niteliklerden bazılarıydı. Dolayısıyla, yazılım geliştirme ciddi bilgi birikimi ve vasıflara sahip, çoğunlukla da akademiyle doğrudan ilişkilere sahip insanlarca yapılıyordu. Fakat, bu niteliklere sahip insan sayısı, zaman içerisinde gösterdiği artışa rağmen talebi karşılayabilmekten uzaktı. Yazılımcılar, doğal olarak, ihtiyaç duyulan küçük bir zümre içerisinde olmanın avantajlarına sahiptiler.

Bu dönemlerde, yazılımcıların yaptıkları işin geneli üzerinde bugüne kıyasla daha fazla denetim sahibi oldukları söylenebilir. Programlamanın nasıl yapılması gerektiğine dair herkesçe kabul görmüş, genel geçer prensiplerin henüz olmayışı, yazılımcılara yaratıcılıklarını sergileyebilecekleri daha geniş bir alan sağlıyordu.

Ürettikleri üzerinde teknik anlamda söz söyleyebilme, karar alabilme otoritesi çoğunlukla doğrudan üretici pozisyondaki yazılımcılara aitti. Dolayısıyla da üretimin gerçekleştirilebilmesi ve devam ettirilebilmesi için yazılımcılara duyulan ihtiyaç ve bağımlılık daha fazlaydı.

Emek gücünün talebe oranla daha az olduğu şartlarda, yazılımcıları bir şirkette ya da organizasyonda tutabilmek zor ve masraflı olabiliyordu. Ayrıca teknik anlamda tek bir kişiye ya da bir gruba bağımlı kalınması üretim'in sürekliliği açısından sermaye sahipleri için tehlikeler barındırıyordu. Çünkü, yazılımcıların bir şekilde üretime devam edememeleri yatırımda bulunan yazılımın tamamiyle boşa gitmesi anlamına gelebiliyordu. Üretim araçlarının gelişimiyle birlikte, teknik üretim süreci standartlarca organize edildikçe, yazılım üretimi üreticilerine daha az bağımlı kabul edilebilecek, daha konvansiyonel bir form aldı. Üretilen yazılım uygulamaları programlama paradigmalarının prensiplerine uygunluğu ölçüsünde okunup görece anlanabilir, nasıl tasarlandığına dair fikirlerin üretilebildiği bir hale geldi. Böylelikle üzerinde çalışılan bir projenin, yazılımcının işe devam etmediği ya da edemediği durumlarda tamamiyle boşa gitmesinin de önüne geçilmiş oldu.

Yazılım geliştirmenin kısaca özetlemeye çalıştığımız tarihsel sürecine bakıldığında, teknik örgütlenmenin üretim ilişkileri üzerinde doğrudan etkilere sahip olduğu çıkarımı yapılabilir. Bakıldığında, kısa tarihçesi içerisinde bile yazılımcıların üretim içerisindeki rolleri kullanılan üretim araçlarına göre farklılıklar göstermiştir. Çalışmamız bağlamındaki rollerinin anlaşılabilmesi için bu üretim araçlarından kısaca bahsedilebilir.

Yazılım geliştirme sürecinin en temel araçlarından birisi yazılımcılar ile bilgisayar arasındaki iletişimi mümkün kılan programlama dilleridir. Bilgisayarlar bu diller vasıtasıyla çeşitli istekleri karşılayabilecek şekillerde programlanabilirler. Bilgisayarlarla iletişimi sağlaması için tasarlanan ve kullanılan dillere bu yüzden programlama dilleri adı verilir.

Modern bilgisayarların ilk yıllarında, programlama ikilik sayı sistemine dayalı makina dili kullanılarak gerçekleştiriliyordu. Makina dili ya da makina kodu olarak da adlandırılan bu dil salt sayısal temsile dayanıyordu. Yani, yazılımcılar programlamayı bilgisayarın aracı bir yazılım uygulamasına ihtiyaç duymadan, doğrudan okuyabildiği makine diliyle yapıyorlardı. Fakat, makina dilini öğrenmek

ve bu dilde programlama yapmak daha zordu, çünkü ciddi seviyede donanım ve matematik bilgisi gerektiriyordu.

Makina dilinin ardından geliştirilen, ikinci jenerasyon programlama dilleri ise görece soyutlama içeren dillerdi. Bu dillerde numerik temsilin yanı sıra bazı alfabetik semboller de kullanılabiliyordu. Yine de, ikinci jenerasyon programlama dilleri, assembly gibi- makina diline yakın özelliklere sahipti. O yüzden, bu jenerasyon diller, donanım'a olan yakınlıkları sebebiyle 'low-level programming language', yani düşük seviye programlama dilleri olarak tanımlandılar.

'High-level programming languages' yani yüksek seviye programlama dilleri olarak tanımlanan üçüncü ve dördüncü jenerasyon diller yazılım tarihi içerisinde ciddi bir kırılmaya sebep oldu. Hem bu programlama dilleri bilgisayarların mimarilerinden ayrı olarak çalışabiliyorlardı, hem de gündelik dilde kullanılan kelimelerin ve sentaks yapılarının programlama için kullanılabilmeleri mümkün hale geliyordu. Bu diller sağladıkları imkanlar sebebiyle düşük seviye dillere göre çalışanları makinaya daha az bağımlı kılıyorlardı ve bu yüzden programcı dostu olarak tanımlanıyorlardı. Yazılımcılar, öğrenmesi daha zor olan, daha kompleks düşük seviye diller yerine, gündelik dilden beslenen bu dilleri daha kolay öğrenebilecek ve daha az hatayla uygulayabileceklerdi. Günümüzde en yaygın olarak kullanılan C, C++, Java, Javascript gibi diller de bu jenerasyon'a dahil örneklerdendir.

Yazılım uygulamalarının yerine getirdiği görevlerin çeşitliliği ve karmaşıklığı arttıkça, yazılım uygulamasının nasıl çalışacağı ve buna bağlı olarak nasıl dizayn edileceği konuları önem kazanmaya başladı. Yazılım uygulamasının hangi durumlarda hangi tasarım ilkelerine göre planlanması gerektiğine dair prensipler geliştirildi. Programın işlemi nasıl gerçekleştireceğini, dolaylı olarak programlamanın da nasıl yapılması gerektiğini belirleyen bu kurallar/prensipler bütününe programlama paradigması adı verildi. Donanımla ve programlama dillerindeki gelişimle ilişkili olarak, spesifik ihtiyaçları karşılayan spesifik paradigmalar üretildi. Prosedürel programlama, Yapılandırılmış Programlama, Nesne-yönelimli programlama ve bildirim temelli bu paradigmalardan bazılarıdır.

Programlama paradigmaları yaygınlaşana kadar, yazılımcılar karşılaştıkları sorunlara çözüm üretme konusunda yaratıcılıklarını, bilgi ve becerilerini kullanma konusunda daha geniş bir alanda söz sahibiydiler. En verimlisi olsun ya da olmasın, kendilerinin olduğunu söyleyebilecekleri çözümler, konseptler geliştirebiliyorlardı.

Fakat, programlama paradigmalarının yaygınlaşması ve kullanımı ile birlikte yazılımcıların zihinsel aktivite gösterdikleri faaliyet alanları bir nebze sınırlandırılmıştır.

Programlama paradigmaları, üretim'in standartlaştırılabilmesi ve modüler hale getirilmesinde büyük pay sahibi olmuşlardır. Üretim standardize edilebildikçe, parçalarına bölünebilmiş, bu parçalar birbirinden bağımsız çalışabilecek yazılımcılara dağıtılır hale gelmiştir. Bu yönüyle paradigmalar yazılım uygulamasının sadece nasıl çalışacağı ve tasarlanacağı üzerinde değil, nasıl üretileceği üzerinde de etkili olmuştur.

Yazılım geliştirmenin teknik örgütlenmesinde oldukça öneme sahip bir diğer üretim enstrümanı, üretimde kullanılan yazılım araçları ve teknolojilerdir. Kullanım alanına göre çeşitlilik gösteren bu araçlar yazılımcıların üretimi gerçekleştirmesine yardımcı olarak kullanılırlar. Kullanılan bu teknolojilerin neredeyse tamamı başka yazılımcılar tarafından geliştirilen yazılım uygulamalarıdır. En fazla kullanılan araçlardan, IDE olarak adlandırılan yazılım geliştirme platformları yazılımcıların kullanımına yönelik bir çok araç barındırır. Yazılımcılar kodlarını bu platformdaki metin editöründe yazabilir, otomasyon ve hata ayıklama araçlarından yararlanarak hata yaptıkları yerleri tespit edebilir ve düzeltebilir.

Bu araçlarla birlikte, yazılımcılar yazdıkları programların düzgün çalışıp çalışmadığına dair hızlı yanıtlar alabilirler. Hataların bazıları bu platformlar tarafından düzeltilebilir. Bu sayede üretimin otomatizasyona tabi tutularak görece basitleştirildiği, dolayısıyla gerçekleştirilme hızında kayda değer bir artış elde edildiğinden bahsedilebilir.

Yazılım geliştirme çözülmesi gereken sorunlarla sıkça karşılaşılan bir süreçtir. Bu problemlerin çözümü, yazılımcılar için bazı durumlarda hayli can sıkıcı boyutlara ulaşabilir. Görüşme yaptığım yazılımcıların tamamına yakını, böyle durumlarda ilk başvurdukları kaynağın internet olduğunu ifade etti. Bu gün internet'te çeşitli platformlarda paylaşılan bilgiler, ya da bu platformlar aracılığıyla gerçekleştirilen yardımlaşma pratikleri yazılımcıların karşılaştıkları problemlerin çözümüne büyük katkı sağlamaktadır. Yazılımcıların gözünde internet'i yazılım geliştirme sürecinin en kritik ve en dostane unsurlarından bir tanesine çeviren de bu özelliğidir.

İnternet ilk yıllarından itibaren yazılım geliştiriciler'in kullanageldiği bir araç olmuştur. İstenilen bilgiye kısa sürede erişilebilmesini sağlayabilmenin ötesinde, ortak bir konu hakkında tartışabilecek, bilgi paylaşımında bulunabilecek toplulukların bir araya gelmesini mümkün kılmıştır. Bugüne geldiğimizde, sadece yazılımla ilgili değil akla gelebilecek her konu hakkında insanların birbirlerinin sorunlarına yardımcı olduğu soru-cevap websitelerinin kullanımı oldukça yaygınlaşmıştır. Yazılım geliştirme üzerine profesyonel ya da amatör olarak ilgilenen yazılımcıların sorunlarına kolektif bir biçimde yanıt aradıkları Stack Overflow, bilgi paylaşımının yapıldığı bu sitelere bir örnek olarak verilebilir.

Fakat bir başka perspektiften bakıldığında, internet, üretim'in kolektivize istihdam ilişkilerinin ise tekilleştirildiği günümüz koşullarının en asli unsurlarından birisi olarak kabul edilebilir. Elbette, yazılımcıların bu platformlar aracılığıyla, gönüllülük ilişkisine dayalı bir şekilde yardımlaşmaları içinde ciddi bir potansiyel barındırmaktadır. Fakat, bu koşullar altında internet üzerinden gerçekleşen bilgi paylaşma pratikleri; yazılım geliştirme pratikleri ve üretim ilişkileri üzerinde de etkili olmaktadır. Verilen işlerin mümkün olan en kısa sürede bitirilmesinin istendiği bir ortamda, internet; yazılımcıların birbirlerine yardımcı olmasından, bilgi paylaşmalarından önce sermayenin talep ettiği hızın kazandırılmasına, üretkenliğin arttırılmasına hizmet etmekte gibi gözükmektedir. Söz konusu üretkenlik artışı, en azından bu şartlar içerisinde, yazılımcıların aleyhine görünmektedir.

Çalışmanın ikinci kısımında, yazılım geliştirilen iş yerlerindeki denetim ilişkilerine odaklanıldı. Yazılımcılar yöneticilerinin yaptıkları işlere doğrudan müdahale ettikleri deneyimlerden yakınarak bahsettiler. Fakat, teknik konular söz konusu olduğunda, bu tarz yönetsel müdahalelerin aza indirgenmeye çalışıldığı gözlemlendi. Yazılımcının verilen sorunu en verimli yoldan çözebileceği yöntemleri kendisinin belirlemesi yöneticiler tarafından da daha verimli bir yol olarak kabullenilmiş gözüküyor. Fakat, böyle bir serbestliğe, proje son tarihleri üzerinden gerçekleştirilen çalışma saatlerinin denetiminin eşlik ettiği söylenebilir. Yazılımcıların yöneticileriyle ek mesai ve ağır iş yükü gibi konularda sıkça problem yaşadıkları görüşmecilerim tarafından dile getirildi. İşçilere çalışma zamanlarını, işlerini proje son tarihlerine yetiştirdikleri takdirde istedikleri gibi organize edebilecekleri serbestliğin tanındığı yönetsel uygulamalara da rastladık. Bu durum çalışanların leyhine gibi gözükebilir. Fakat aynı durum yazılımcıların kendilerine atanan işleri tamamiyle kendi sorumlulukları olarak görmeleriyle de neticelenebilmektedir. Yani, çalışanlar iş yüklerinin fazlalığını, zorluğunu hesaba katmak-

tansa, işleri yetiştirememeyi kendi başarısızlıkları olarak değerlendirme eğiliminde olabilmektedirler. Bu yüzden işleri proje son tarihlerine yetiştirebilmek adına yöneticilerinin zorlaması olmaksızın fazla mesai yapar hale gelebilmektedirler. Kendine özgü özelliklerine karşın, yazılım geliştirilen üretim yerlerindeki denetim ilişkilerinin artık emek değer birikimini arttıracak ve devamlı kılacak doğrultuda şekillendiği iddia edilebilir.

Son bölümde, yazılımcıların geleceklerinin belirsiz oluşu sebebiyle deneyimledikleri iş güvencesizliği; üretim için ihtiyaç duyulan vasıflarla kurulan ilişkilerle birlikte ele alınmıştır. Sektörün kısa bir tarihe sahip olması, üretim organizasyonunun hızlı dönüşümü yazılım sektöründe çalışanların akıllarında, en azından yazılım geliştirmenin teknik kısmında yer alanlarda gelecek kaygısı yaratabilmektedir. Sektörün ne tür dönüşümlere tanıklık edeceğinin belirsizliğiyle birlikte, yaşlanmanın bugünün iş dünyasındaki olumsuz çağrışımları bu kaygıları belli bir yaştan sonra güçlendirmektedir. Bu durum, yazılımcılara gelecek anlamında pek az seçenek tanır, yönetici pozisyonuna geçmek veya üretimin gerektirdiği veya gerektirebileceği vasıflara sahip olmak. Gözlemlenildiği kadarıyla, yazılımcılar deneyimledikleri iş güvencesizliğini bu stratejilere başvurarak çözme eğilimindedirler. Çalışılan farklı iş yeri sayısının, yazılımcılar arasında oldukça yüksek rakamlar göstermesi bu eğilime bir kanıt olarak düşünülebilir. Fakat, emek piyasasındaki rekabet arttıkça, üretim için gerekli emeğin yeniden üretimi de görece ucuzlar. Bu bağlamda kazanılan üretkenlik artışının, bugünün istihdam etmeden büyüme stratejisi güden dışlayıcı bölüşüm politikaları içerisinde işçilere daha güvencesiz ve daha rekabetçi çalışma koşulları olarak geri döneceği söylenebilir. Bu yüzden yazılımcılar, bireysel ve piyasa güdümlü stratejiler ile çözüm aradıkları güvencesiz ve belirsiz çalışma şartlarının yeniden üretiminin bir parçası haline gelirler.

APPENDIX F: TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü ☐

Sosyal Bilimler Enstitüsü ☐

Uygulamalı Matematik Enstitüsü ☐

Enformatik Enstitüsü ☐

Deniz Bilimleri Enstitüsü ☐

YAZARIN

Soyadı :
Adı :
Bölümü :

TEZİN ADI (İngilizce) :

TEZİN TÜRÜ : Yüksek Lisans ☐ Doktora ☐

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
3. Tezimden bir bir (1) yıl süreyle fotokopi alınamaz. ☐

TEZİN KÜTÜPHANEYE TESLİM TARİHİ: