

**The London School of Economics and Political Science**

**An Institutional Analysis of the Formation of Jobs in Software Work in the  
United States, 1945-2001**

Roohollah Honarvar

A thesis submitted to the Department of Management of the London School of  
Economics for the degree of Doctor of Philosophy, London, April 2015

## **Declaration**

I certify that the thesis I have presented for examination for the PhD degree of the London School of Economics and Political Science is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it).

The copyright of this thesis rests with the author. Quotation from it is permitted, provided that full acknowledgement is made. This thesis may not be reproduced without my prior written consent.

I warrant that this authorization does not, to the best of my belief, infringe the rights of any third party.

I declare that my thesis consists of 95,307 words  
(including footnotes but excluding bibliography and appendices).

## **Abstract**

Information technology (IT) jobs are characterized by continuous change and increasing fragmentation, yet there is no explanation for why this has been the case. In this research, drawing on a historical institutional perspective, I study IT work as an instant of skilled work under capitalism that is subject to interactions among different institutions over long periods of time. In particular, I focus on the dynamics of the relationship between the institutional arrangements of skill formation in IT work, competition in the IT industry and employment relationships in the IT labor market. The main actors involved in the shaping of these institutional arrangements are the government, IT corporations and workers.

Empirically, the research focuses on the history of software work in the U.S. IT industry (1945-2001). I draw on existing histories of the industry and corporations as well as a variety of publications that shed light on the industry, individual corporations, training institutions or the labor market. This includes newspapers and magazines, professional and trade journals, government and industry reports, company reports and archives, oral histories and biographies of software workers or businessmen. I study four periods of the history of the IT industry: the emergence of software work and the rise of IBM (1945-1954), IBM's dominance before the unbundling decision (1955-1969), the dominance and decline of IBM (1969-1985), and finally, the rise of Microsoft until the burst of dotcom bubble (1985-2001). The central argument of this thesis is that, in the absence of a governmental intervention aimed at stabilizing the labor market, the strategic and tactical decisions of IT corporations involved in the dynamics of IT product competition destabilized IT jobs and effectively prevented the formation of a solid occupational collectivity which is a precondition of institutionalization of jobs.

## Acknowledgements

This work would have not been possible without the constant support of my parents who patiently watched over their son to pass through the various stages of academic life. There is no way I can thank them enough. I am forever grateful.

My childhood sweetheart, love of life and dear wife, Samaneh, has been a constant source of encouragement and support. She has been the true ‘programmer’ of our life, without whom I would have not been able to accomplish this work. She and our sweet daughter, Salva, made home such a comfortable place that I did not need to go anywhere else to work on this thesis.

I also thank my sister and brother, Dorri and Rooholamin, who in spite of the distance between us, were always there whenever I needed them.

I have learned a lot from the intellectual environment of ISIG and I am grateful for the opportunity. In particular, I wish to thank Jannis Kallinikos, my second supervisor, for his constant support and invaluable advice. Carsten Sørensen and Will Venters also provided valuable feedback at earlier stages of my research; I thank them both.

My virtuoso friend, Cristina Alaimo introduced me to *Canvases and Careers*, a book that shaped my approach in this study. Grazie! I have also enjoyed the conversations and discussions I have had with Anouk Mukherjee and Mahmood Zarger, my friends in this long journey in the IS field. Thank you guys.

Finally, I am indebted to my supervisor Prof. Chrisanthi Avgerou. She was my teacher before I came to LSE and will remain my mentor long after I leave. With her generosity, patience and kind advice, she taught me lessons that I will not forget.

## Table of Contents

List of abbreviations	8
List of tables	10
List of figures	11
1 Introduction.....	12
1.1. Background	12
1.2. Statement of the problem	15
1.3. Structure of the text	18
<b>Part One: Literature, Theory and Methodology</b>	
2 Literature Review.....	21
2.1. Introduction	21
2.2. Understanding occupations and occupational change	21
2.3. Understanding occupational change in IT work	23
2.4. Occupational change in the context of IS departments	27
2.5. Summary and conclusion	28
3 Definitions and Theoretical Framework.....	30
3.1. Introduction	30
3.2. Occupations and skills in the age of organizational dominance	32
3.3. Linking regimes of skill formation to competition in the IT industry	36
3.4. Placing the theoretical framework within institutional theory	39
3.5. Conclusion	43
4 Methodological Considerations and Notes on Historiography.....	45
4.1. Background	45
4.2. Process tracing and colligation in historically-oriented social research	47
4.3. Reflections on my research	51
4.4. The ontological and epistemological foundations of my research	54
4.5. Constructing the objects of study	56
4.5.1. <i>The object of study: software work</i>	56
4.5.2. <i>The geographical boundaries</i>	60
4.5.3. <i>The beginning, the end and turning points</i>	61
4.6. Notes on the sources	62
<b>Part Two: Historical Narrative – Software Work in the American IT industry 1945-2008</b>	
5 The Rise of Software Work: 1945-1954.....	65
5.1. Introduction	65
5.2. The Beginning: proto-software work in two early cases	65
5.3. Analysis	68

5.4. Into 1950s: Farewell Mathematics, Hello Automated Programming	71
5.5. Analysis	82
5.6. The origins and legacy of SAGE project	91
5.7. Analysis	97
5.7.1. <i>Psychometric tests</i>	97
5.7.2. <i>The software factory ideal and the organization of software development</i>	102
5.8. Summary	105
6 The Rise of IBM in the Computer Industry: 1954-1969.....	107
6.1. The shaping of competition in the computer industry	107
6.2. Analysis of the competition	121
6.3. The importance of vendor-provided training	125
6.4. Rapid expansion of the labor market	132
6.4.1. <i>The socio-political environment of labor market</i>	137
6.4.2. <i>Associations of software workers</i>	139
6.5. Employment policies	142
6.6. Implications for software work	146
7 An IBM World: 1969-1985.....	150
7.1. Competition in the industry	150
7.2. Analysis of the competition	160
7.3. The role of dissemination of skills in the transformation of microcomputers	161
7.4. Varieties of Software Work	168
7.4.1. <i>The Computer industry</i>	169
7.4.2. <i>Software companie</i>	176
7.4.3. <i>Analysis</i>	178
7.5. Skills and the boundaries of labor market	181
7.5.1. <i>Continual problematic of skills</i>	181
7.5.1. <i>Status of programmers according to the courts and government</i>	181
7.5.2. <i>SCDP and the Licensing of Data Professionals</i>	184
7.6. Implications for software work	186
8 New competition, new workplace: 1985-2001.....	188
8.1. The rise of Microsoft	188
8.2. Analysis of the competition	193
8.3. Employment Policies	195
8.3.1. <i>Employment policies and decisions in IBM</i>	196
8.3.2. <i>Other computer companies</i>	201
8.3.3. <i>Software companies</i>	202
8.3.4. Employment policies and decisions at Microsoft	204

8.4. A new institutional arrangement for vendor-provided training	208
8.5. Labor market	218
8.5.1 <i>Worker associations and employee groups</i>	219
8.5.2 <i>The role of IT companies and business associations in shaping the IT labor market</i>	223
8.6. Implications for software work	228

### **Part Three: Discussion and Conclusion**

9. Discussion.....	236
9.1. Summary of the narrative	236
9.2. Reconstructing the argument	243
9.3. Developments since 2001	249
10 Conclusion.....	253
10.1. Summary of the argument	253
10.2. Placing IT work in its social and historical context	254
10.3. Learning lessons	256
10.4. Thinking about the future	257
10.5. Contributions	260
10.6. Limitations and further research	260
Appendix: Event map of the narrative.....	262
References.....	267

## List of Abbreviations

ACM	Association of Computing Machinery
ACPA	Association of Computer Programmers and Analysts
ACT	Advanced Computer Techniques
ADAPSO	Association of Data Processing Service Organizations
ADP	Automatic Data Processing
ADR	Applied Data Research
AMD	Advanced Micro Devices
AOL	America Online
ASK	ASK Group
BYU	Brigham Young University
CASE	Computer-Aided Software Engineering
CCP	Certificate in Computer Programming
CDC	Control Data Corporation
CDP	Certified Data Professional
C-E-I-R	Council for Economic and Industrial Research
CNE	Certified Netware Engineer
CNP	Certified Network Professional
CompTIA	Computing Technology Industry Association
CPA	Certified Public Accountant
CSC	Computer Sciences Corporation
CUC	Computer Usage Company
DEC	Digital Equipment Corporation
DPMA	Data Processing Management Association
EDP	Electronic Data Processing
EDS	Electronic Data Services
EMCC	Eckert-Mauchly Computer Corporation
ERA	Engineering Research Associates
FLSA	Fair Labor Standards Act
GE	General Electric
GEISCO	General Electric Information Services
HP	Hewlett-Packard
IBM	International Business Machines
ICCP	Institute for Certification of Computer Professionals
ICDL	International Computer Driving License
IDC	International Data Corporation
IEEE	Institute of Electrical and Electronics Engineers
ISP	Internet Service Provider
ITAA	Information Technology Association of America



IWA	International Webmasters Association
MCP	Microsoft Certified Professional
MSA	Management Science America
MSDN	Microsoft Developer Network
NCR	National Cash Register
NMAA	National Machine Accountants Association
NPA	Network Professionals Association
NRC	National Research Council
NWCET	National Workforce Center for Emerging Technologies
PAT	Programmer Aptitude Test
PATCO	Professional Air Traffic Controllers' Organization
RAND	RAND Corporation
RCA	Radio Corporation of America
SBC	Service Bureau Corporation
SCDP	Society of Certified Data Processors
SDC	System Development Corporation
SDD	System Development Division, RAND Corp.
SDK	Software Development Kit
TRW	Thompson, Ramo and Wooldridge Inc.
UCC	University Computing Company
WashTech	Washington Alliance of Technology Workers

## List of Tables

Table 3-1: Work structures (adapted from Kalleberg and Berg, 1987).....	31
Table 3-2. Three types of employment systems (adapted from Fligstein, 2001).....	33
Table 3-3. Varieties of Institutionalism.....	42
Table 3.4. Elements of the theoretical explanation.....	44
Table 4-1: Elements of the explanatory narrative.....	52
Table 5-1: Estimates of the different kinds of workers required in a 1954 study.....	78
Table 6-1: Sectors of the software and services industry.....	115
Table 7-1 Share of each sector of the computer industry of the market.....	160
Table 7-2 Pioneering software PC entrepreneurs.....	165
Table 7-3 Worldwide number of IBM employees, 1964-1972.....	170
Table 8-1: Number of IBM U.S. employees, 1986-2001.....	199
Table 8-2: Layoffs at Apple, 1980-2001.....	202
Table 8-3: Number of Novell employees between 1993 and 2001.....	203
Table 8-5: Level of cap and number of H1-B visas issued between 1992 and 2003.....	228
Table 8.6: Fragmentation of software work in the 1990s.....	234
Table 9-1: Number of Microsoft employees, 2006-2012.....	251

## List of Figures

Figure 2-1: The ‘static picture’ of changes in computer systems development (Friedman and Cornfield, 1989)	27
Figure 3-1: The role of skills in the relationships between various social groups who have an interest in taking over a task area	32
Figure 3-2: Relationships in the theoretical model.	43
Figure 5-1: IBM programmer ad, 1952	72
Figure 5-2: IBM advertisement for Business Analysts, 1956	77
Figure 5-3: Honeywell’s ad for ‘Computer method analysts’, 1959.	79
Figure 5-4: IBM’s programmer ad, 1956	81
Figure 5-5: General Electric’s ad for programmers. 1959	84
Figure 5-6: Sperry Univac ad for new computer, New Yorker, 2 Oct. 1978	90
Figure 5-7: Rand’s programmer ad, 1954.	92
Figure 5-8: SDC’s programming ad, 1957	94
Figure 6-1: Univac’s ad. 1955	109
Figure 6-2: IBM’s Service Bureau Corporation’s ad, 1957.	112
Figure 6-3: RCA’s ad for programmers, analysts and sales representative. 1959	124
Figure 6-4: IBM’s ad for attracting programmer trainees, 1959	126
Figure 6-5: IBM’s ad for ‘Applied Science Representatives’, 1954	128
Figure 6-6: IBM’s ad for programmers, 1954.	129
Figure 6-7: CDC advertisement for programmer trainees, 1968.	131
Figure 6-8: IBM’s ad for college graduates, 1961	133
Figure 6-9: Two sample ads from EDP schools, 1967	135
Figure 6-10: IBM’s programmer ad in <i>Ebony</i> magazine, 1968	138
Figure 6-11: IBM’s programmer ad in <i>Ebony</i> magazine, 1968	145
Figure 8-1: Microsoft University ad, 1988	212
Figure 8-2: Employment in U.S. IT industries, 1990, 2003	229

# 1 Introduction

"To be 'redundant' means to be supernumerary, unneeded, of no use - whatever the needs and uses are that set the standard of usefulness and indispensability. The others do not need you; they can do as well, and better, without you. There is no self-evident reason for your being around and no obvious justification for your claim to the right to stay around. To be declared redundant means to have been disposed of because of *being disposable* [...]" (Bauman, 2004, emphasis in the original)

## 1.1. Background

In the morning of March 12th, 2008, Bill Gates appeared before the US Congress' Committee on Science and Technology to tell them about "the shortfall of scientists and engineers with expertise to develop the next generation of breakthroughs":

**“MR. ROHRABACHER:** There are a lot of other people in the society rather than just the top people.

**MR. GATES:** That's right.

**MR. ROHRABACHER:** It's the B and C students that fight for our country and kept it free so that people like yourself would have the opportunity that you've had. Those people, whether or not they get displaced by the top people from another country is not our goal. Our goal isn't to replace the job of the B student with A students from India.

**MR. GATES:** That's right. And --

**MR. ROHRABACHER:** And the B students deserve to have good jobs and high-paying jobs.

**MR. GATES:** That's right. And what I've said here is that when we bring in these world-class engineers, we create jobs around them.

**MR. ROHRABACHER:** Okay.

**MR. GATES:** And if we don't -- so the B and C students are the ones who get those jobs around these top engineers. And if these top engineers are forced to work, say in India, we will hire the B and C students from India to work around them.

**MR. ROHRABACHER:** Okay. But, according to "BusinessWeek", there are almost 150,000 programmers [who] have lost their job in this country since the year 2000. Now my reading of all of this is that there are plenty of people out there to hire, but people want to have the top-quality people from India and China and

elsewhere, and they're willing to let these 150,000 American computer programmers just go unemployed.

**MR. GATES:** Actually, "BusinessWeek" doesn't do surveys. I think you're referring to a quote in "BusinessWeek" from an Urban Institute study --

**MR. ROHRABACHER:** That's what I said, according to "BusinessWeek", yeah.

**MR. GATES:** Well, they quote -- it's not according to "Business Week".

**MR. ROHRABACHER:** Okay.

**MR. GATES:** There was a study that a group at Urban Institute did that was deeply flawed in terms of how it defined what an engineer is. When we say that these jobs are going begging, we're in business every day.

**MR. ROHRABACHER:** Mm-hmm.

**MR. GATES:** We're not kidding about it. These jobs are going begging, and the result is that in a competitive economy --

**MR. ROHRABACHER:** You'd have to raise wages.

**MR. GATES:** No, no --

**MR. ROHRABACHER:** -- if the job is going begging, you raise wages, now in --

**MR. GATES:** No.

**MR. ROHRABACHER:** Okay.

**MR. GATES:** It's not an issue of raising wages. These jobs are very, very, very high-paying jobs.

**MR. ROHRABACHER:** Okay --

**MR. GATES:** And we are hiring as many of these people as we can.

**MR. ROHRABACHER:** Let me give you one example --

**CHAIRMAN GORDON:** Mr. Rohrabacher, if you don't mind, we'll finish this on the second round.

**MR. ROHRABACHER:** You know, I am one of the guys that helped Kosovo become independent, and I'm on the Foreign Relations -- hearing there. Maybe at the reception tonight, which you're going to be at, maybe we can continue this discussion.

**CHAIRMAN GORDON:** I'm sure [Mr. Gates] will be excited to know you'll do that.

(Laughter)" (Microsoft, 2008)<sup>1</sup>

---

<sup>1</sup> Microsoft (2008) 'Bill Gates: Testimony before the Committee on Science and Technology, U.S. House of Representatives', <http://news.microsoft.com/2008/03/12/bill-gates-testimony-before-the-committee-on-science-and-technology-u-s-house-of-representatives/> published 12 Mar 2008, accessed 30 Jun 2011.

As Mr. Gates would admit, contrary to the popular images of ‘computer geeks’ and a history full of genius inventors and entrepreneurs, “there are a lot of other people” involved in the so-called information technology (IT) revolution. Unlike “the top people”, however, we know very little about the jobs and working lives of those who carried out much of the actual work; i.e. those who have designed, developed, operated and maintained information and communication technologies across the new “information society” (Blok and Downey, 2003; Webster, 2006;). This brief exchange, by far the most heated part of the day, is illustrative of their situation in the society: they are everywhere but nowhere to be seen. They work with information, but there is very little information about them. One cannot be sure about what happened to those “150,000 programmers”; unlike Bill Gates, they are only represented in abstract and debatable figures.

At the heart of the matter, as Gate’s skillful twist on the issue shows, is how to define what an engineer (or any other IT job) is. Recent research suggests that it is by no means clear what constitutes an IT job and who the IT workers are (Kaarst-Brown and Guzman, 2005). This is perhaps most evident in the huge differences that exist between various estimates of the number of IT workers: between just under 2 million to over 5 million in the year 2000 (see National Research Council (NRC), 2001: Appendix B). Nor it has been possible to classify IT jobs within any stable system of occupational classification. According to the U.S. NRC Committee on Workforce Needs in Information Technology:

‘Definitions of data categories (e.g. the definitions for items such as “computer scientist” or “systems analyst”) which are necessarily relatively stable over time, do not necessarily reflect the IT job titles of today because new types of jobs emerge quickly in the IT sector. In particular, data categories are too coarse and do not reflect important distinctions between IT jobs.’ (NRC, 2001: xvi)

Even certain terms like ‘programmer’ or ‘systems analyst’, which have been continuously used for the past 60 years, now refer to very different jobs with a diverse range of roles and responsibilities, requiring different sets of skills and forming part of different career paths (NWCET, 2003).

Closely related is how one should differentiate between the so-called “world-class engineers” around whom jobs are supposedly created and the normal (‘B or C class’) programmers. This is not simply a matter of job evaluation and wage differentials. It rather concerns, more fundamentally, how the constitution of skills, their formation in individuals

(training) and the allocation of responsibilities and rewards that go with them are decided.<sup>2</sup> Jobs emerge when an activity is acknowledged as a form of work that can be assigned to individuals and is worthy of remuneration. Occupations arise when these local acknowledgements become more widespread and new institutional arrangements emerge to structure their work and shape their relationships with employers, etc. (Nelsen and Barley, 1997). Often at the center of various forms of institutional support is the provision of training and protection of individuals' skills (licensure in the case of professions, retraining rights in union contracts). But in the absence of stabilized IT jobs, one can hardly expect to find occupationally defined skills and skill levels. This raises the question why are IT jobs continuously changing and differentiating. How is IT work related to the wider institutions, especially those concerned with formation, recognition and utilization of skills' in workers. Is there something special about IT work, or are other jobs and the way working life is organized undergoing a more fundamental change? And is there a link between the two?

The conversation above also shows that the supply of IT workers is a cause of concern for policy makers. IT workers play an increasingly vital role in the economy (Freeman and Soete, 1994) and governments have recognized the need for devising policies that help building an IT workforce (e.g. The Twenty-First Century Workforce Commission Act in U.S.). Much of these efforts have been driven by a perceived shortage of IT workers as well as a belief in the potential of IT for job creation and economic growth. Governments want to know what skills to invest in, and how best to train and retain IT workers. Lack of understanding of how and why IT jobs change has made their plans for training IT workers problematic (NRC, 2001).

## **1.2. Statement of the problem**

In this study, my primary question is why IT jobs are not institutionalized in the way many other occupations have been? The first answer that springs to mind is "because of constant technological change". But it seems too simplistic to assume that technological change is the primary cause of differentiation and fragmentation of IT jobs. There are other occupations that have been subject to technological change but show no indication of significant change (see Gallie, 1991). Moreover, it is people in IT jobs who often bring about technological change in IT. The question then, would be why (and how) people do this to their jobs. At the very least, a satisfactory explanation must unpack this relationship and place it within its social context. More fundamentally, one could argue that taking technology as the primary cause reduces jobs to their material aspects whereas jobs are social constructions that change

---

<sup>2</sup> By skill I mean the capacity to accomplish a task (cf. Freidosn, 2001: 25). It may be gained through experience or education and usually has productive value (see Green, 2013: Ch. 2).

meaning and significance as a result of changes in the social environment (see Barley, 1988). Therefore one must look for alternative and more complicated explanations beyond a simple technological one.

The unstabilized status of IT jobs is interesting for other reasons as well. Historical and sociological studies of professions have shown that when a group of workers have skills that are highly valued (or in high demand), they create a monopoly of skill and raise barriers to the entry of 'others' (e.g. Larson, 1977; Perkin, 1989). Given the persistent shortage of IT workers, which started very early in the history of the field, one could hypothesize that they could, in principle, create such a monopoly and control the emerging labor market in order to establish their positions and privileges in the society and across organizations. But this never happened: IT workers seemed powerless (if not care-free) in exerting any influence over their labor market or even changes in their jobs. Instead, IT became an enabler for ever-expanding businesses that, in the process of globalization, crossed all borders in search of cheap labor or valuable skills. As Mr. Gates pointed out "in a competitive economy", "if these top engineers are forced to work, say in India, we [in the industry] will hire the B and C students from India to work around them."

To be sure, professionalization of IT work has been high on the agenda of IT workers and academicians for a long time (e.g. Orden 1967; Denning, 2001, Ensmenger, 2001a). They have produced an extensive knowledge-base, established professional bodies and developed elaborate higher education programs. Some groups have also made extensive efforts to establish a scientific 'objective' basis for the profession under such banners as 'Computer Science', 'Software Engineering' or even the Capability Maturity Model (CMM). Others, not being content with a purely technical education, have emphasized the social and ethical aspects of IT workers' role (e.g. Myers, Hall and Pitt, 1996; Walsham, 1996; Dahlbom & Mathiassen, 1997; Kling, 2003). Yet all these efforts seem to have had little effect on the rapidly changing world of IT work: professional bodies suffer from very low membership rates while educational programs witness widening of the gap between theory and practice (Abbott, 2005; Iivari, Hirschheim and Klein, 2008).

The academic Information Systems (IS) departments and programs are probably the epitome of the effects of the continuously changing mix of jobs and skills. On the one hand, IS scholars have asked themselves (over and over) whether or not IS is a discipline and has a core, and if so, what is it? (See, among others, Banville and Landry, 1989; Paul, 2002; King and Lyytinen, 2006; Somers, 2010). The answers to these questions have often been (implicitly or explicitly) related to what IS scholars should teach (e.g. Markus, 1999; King



and Lyytinen, 2004). On the other hand, cyclic fluctuations in the enrolment figures of IS programs have sparked discussions about the credibility of IS programs and relevance of its teachings to practitioners 'real' concerns and needs (e.g. Trauth and Hafner, 2000; Glass, 2007; Looney and Akbulut, 2007; Panko, 2008; Bullen et al., 2009, Riemenschneider et al., 2009; Davidson, 2011; Firth et al., 2011; Looney et al., 2014). But despite all their efforts to be academically respectable and practically relevant, IS departments seem to be losing ground in both areas.<sup>3</sup>

All these issues point to the necessity of understanding why IT jobs have been continuously changing and fragmenting. It must be noted that, as indicated above, IT jobs have nevertheless been shaped by the institutional context within which they emerged and evolved. It is my contention that the specific context of emergence of IT work in the US played a significant role in setting the trajectory of IT jobs. More precisely, I argue that, in the absence of governmental intervention aimed at stabilizing the labor market, decisions made and policies adopted by IT corporations involved in the dynamics of product market(s) have led to destabilization of IT workers and effectively prevented the formation of a solid occupational collectivity. In the absence of lineages (organizational and occupational careers and connections that bond workers together), jobs became transitory in an essential sense: their practices, their role and responsibilities, all became subject to rapid and seemingly uncontrollable change with deteriorating effects for their claims to authority and legitimacy.

How did this happen? It was surely not an automatic outcome of changes in the technological component of jobs. As I seek to demonstrate in this research, for a period of time, despite rapid technological change, jobs and careers seemed to be stabilizing, largely thanks to favorable corporate HR policies that sought to retain valuable skills in individuals. However, with the decline of IBM and transformation of competition in the IT market, the situation changed: Not only new skills came to be valued but also the new dominant companies felt no responsibility towards training individuals or keeping skilled individuals. The new norms of employment relationship that emerged in the wider economy allowed corporations to feel free from any obligation to provide employment security (or even training) and therefore, skills became the responsibility and liability of individuals. Throughout both periods, the effectiveness of corporate decisions (in setting the terms of employment and dealing with workers) was largely due to government policies that lent support to corporations rather than workers.

---

<sup>3</sup> A study of recent college graduates after the financial crisis had found that 14.7 percent of IS majors were unemployed. The average unemployment rate of the group of graduates studied was 7.9%. See Kirkwood, L (2013) 'Art majors jump ahead of tech grads in landing jobs', *USA Today*, 30 Jul, 2013.

Moreover, IT workers became the harbingers of the new forms of work and IT jobs were considered a 'model' for future white-collar jobs (e.g. Kanter, 1995). In the new world of work, corporations can (are allowed and powerful enough) to set the standards of 'skill requirement' without having any commitment to provide those skills or to retain and retrain skilled employees. In the new employment relationship, the onus is on the (isolated) individual to show his/her continued worth. Thus, according to Castells (2001: 93), the new 'informational society' demands 'self-programmable labor', that is, "anyone with the capacity to develop skills as required by a changing market" and will pay well for those 'talents' (see also Castells, 2010: xxiii). But, whether this reflects arbitrarily set skill 'requirements' or fluctuations in the market, the new political economy of skills has made even the best jobs transitory. Once those skills are expired, lives spent learning the "wrong" skills are considered a form of 'human waste' that must be discarded. Unless they engage in 'life-long learning', those deemed redundant by "the market" will find themselves not simply unemployed but increasingly 'unemployable' (Bauman, 2004, p. 5, 12-14). This combination of precarious and precious work has created a job market in which skill mismatch is found side-by-side polarization within and across occupations (Kalleberg, 2007, 2011).

### **1.3. Structure of the text**

The historical orientation of this study has made it necessary to divide this thesis into three parts. The first part includes literature review, theoretical framework and a description of the methodology used. In the first chapter of this part, chapter two, I will review the literature on occupational change and the attempts that have been made to understand changes in IT jobs. After showing that none of the existing explanatory frameworks can explain the situation of IT jobs, I will present, in chapter three, the theoretical framework of my research. Built on institutional theory, the theoretical framework presented focuses the study on interactions among actors in different institutional domains over extended periods of time. This necessitates a historical study, which its methodological underpinnings as well as ontological and epistemological principles are described in chapter four.

In part two, which consists of four chapters, I will present the empirical product of this study, i.e. a historical narrative of how developments in different institutional domains have shaped the formation of software jobs in the U.S. IT industry between 1945 and 2001. Beginning with the rise of software work before the formation of competition, I will discuss the influence of three major early developments, i.e. the commercialization of computers, the development of programming languages and other tools, and the role of the System

Development Corporation in training an early generation of programmers. Each of the following three chapters covers one episode of a continuous narrative that begins in the first half of 1950s and ends in 2001. In each chapter, I begin with a description of competition, followed by an analysis of its changes in the episode concerned. Then I consider the conditions of employment relationships and training in the labor market and their role in the competition and/or the implications of competition for them. I will particularly focus on the attempts towards formation of worker collectivities or controlling the labor market by workers and the role played by the government. In my discussions, I will demonstrate how the progressive variety of types of software work can be understood in terms of changes in the structure of industry and the relationships between workers, employers and the government in different domains.

In part three, I discuss the main findings of the study. In the first chapter of this part, chapter nine, I provide a summary of the findings of this research and present them in a consolidated form. I will then use my analysis to shed light on the developments that have occurred in IT work since 2001. In the last chapter, I summarize the argument presented, discuss its limitations and contributions and point towards the possible directions of future research.

**Part 1**  
**Literature, Theory and Method**

## **2 Literature Review**

### **2.1. Introduction**

While sociology of work has expanded in many directions in recent years, relatively few theoretical perspectives have been developed that aim at understanding changes in occupations. Indeed, until 1970s, the majority of sociologists assumed occupations to be “well-defined, constant occupations, with constant statuses” (Abbott, 1993: 189). Similarly, while there is an abundance of situated studies of IT work, there has been little attempt at providing a coherent analysis of changes in IT occupations (for an early critique, see Orlikowski, 1988; Orlikowski and Baroudi, 1989). Therefore, in this chapter, I first review the sociological literature on occupations and occupational change and then examine the attempts that have so far been made to understand changes in IT occupations. As I will show, none of these attempts have been able to capture the essence of changes that are occurring in IT jobs and therefore cannot provide a satisfactory explanation for them. Then I will consider the only systematic study of changes in IT jobs, namely, Friedman and Cornford’s (1989) study of evolution of IS departments in corporations. I will show that despite producing an impressive range of insights about changes in IT jobs, Friedman and Cornford’s focus on internal workings of IS departments has left out important institutional elements. I conclude by submitting that these alternative explanations must take into account multiple aspects of IT work from an institutional perspective and historical outlook.

### **2.2. Understanding occupations and occupational change**

Our common-sense understanding of occupations and much of the traditional sociology of occupations is based on an implicit recognition of their institutionalized characteristics (e.g. Dunkerley, 1975). Even though sociologists have long recognized that occupations are evolving structures and, as such, are subject to constant amalgamation and differentiation, there has been relatively little progress in our understanding of how this happens (Turner and Hodge, 1970; Miller, 1988). One reason for this is that occupational cohesion is a multifaceted issue and a range of factors can influence it: while occupations are usually recognized as social groups made up of people doing a similar kind of work, the resemblance among their task areas may be defined by the tools they use, clients they serve, products they produce, organizational positions they occupy, etc. Moreover, other properties such as similar experiences, skills or credentials, workplace location and culture, or race and gender backgrounds can have an effect on the way they are identified and distinguished (Abbott, 1991a). There is no necessary causal relationship between any of these elements

and therefore the trajectory of formation of one occupation can be very different from another.

It is therefore not surprising that the process of institutionalization of jobs is a multifaceted phenomenon (Miller, 1988): when a set of jobs with certain roles and responsibilities became widespread and relatively stabilized, they share (or are governed by) similar cognitive, normative and regulative frameworks, use similar resources or techniques and occupy similar social positions. Their claims to legitimacy and authority are based on similar and socially recognized foundations. More or less standardized job profiles come to be used by work organizations in their recruitment and compensation processes across industries. More formally, similar jobs form a distinct category recognized as an occupation in national schemes of occupational classification (see Zucker, 1987: 455-6). Jobs related to an occupation can even crystallize into “orderly sequenced” occupational careers (Slocum, 1974).

In the pre-industrial period occupations corresponded to the unit of family (rather than individuals), which in turn placed the family in a class (e.g. peasants worked on farms while noblemen had no occupation; Tilly and Tilly, 1994). The emergence of factories as the primary providers of products and services (replacing artisan workshops and family businesses) was the distinctive feature of industrialization. Individuals, in turn, were directed to offer their skills and capabilities to work organizations through ‘labor markets’. Over time, the concept of occupation came to be identified with the structural similarities of tasks and responsibilities of individual jobholders. In order to make sense of their population, governments began to gather data and organize different groups of jobs into classified and recognizable occupations (see Katz, 1972; Davies, 1980; Anderson, 1990).

But beyond mere categories, occupations have existed (at least for the past few centuries) as social groups. These social groups were formed to protect the common interest of members (jobholders) against negative external influence from rival groups, employers, governments, etc. (Volti, 2008). Also, in order to endure, they created mechanisms for retaining and reproducing their skills, roles and responsibilities in workplace, boundaries and relationships with others, etc. This has been the historical and traditional role of guilds, unions and professional associations (Rueschemeyer, 1986; Krause, 1996). Many sociologists have therefore tried to understand the mechanisms of occupational control. Most theories of occupational control emerged within the body of sociology of professions, while other (lower-status) forms of work were perceived to be under external control (for a review, see

Simpson, 1985). In the rest of this chapter, I will review the various attempts that have been made to understand changes in IT work using this literature.

### **2.3. Understanding occupational change in IT work**

Within the framework provided above, IT jobs can be defined as the set of jobs which are mainly responsible for developing and deploying IT as part of an employment relationship. Alternatively, IT workers are those individuals who occupy these positions and assume these responsibilities.<sup>1</sup> Accordingly, evidence for the non-institutionalized status of IT jobs can be found in the frequent change and reclassification of these jobs in occupational classification schemes, the absence of professional associations or unions that represent the interests of workers or regulate entry of individuals to the occupation, the diversity and fragmentation of practices of jobholders who seemingly occupy similar jobs and the variety of employment relationships and career paths of IT workers (Pettigrew, 1973; Hebden, 1979; cf. Kaarst-Brown and Guzman, 2005).

The study of changes in computer occupations began in the late 1970s when researchers turned their attention to changes in the jobs of computer workers (Kling and Gerson, 1977; Anderson and Mortimer, 1979). Some authors (such as Kraft, 1977, Greenbaum, 1979) tried to understand changes in IT occupations as a case of the ‘deskilling thesis’ put forward by Braverman (1974). In his influential analysis of work in the twentieth century, Braverman (1974) suggested that capitalists and managers were increasingly using technologies to rationalize work, deskill jobs and change the structure of occupations. He identified extreme specialization, routinization, and automation as three mechanisms through which jobs were being deskilled. While most of the subsequent literature focused on examining the so-called ‘de-skilling thesis’, he was clear in his analysis that in practice all such effects will be limited, not least because the deskilling process itself “brings into being new crafts and skills and technical specialties” (p.172). To Braverman, the advance of electronic data processing machines was clear evidence of this trend, but nevertheless he argued that ultimately “data processing occupations” will be among the first victims of deskilling process (p. 227-239). While studies from this perspective can throw light on the categories of IT work that have been eliminated (like coders) it is clear that IT jobs (as a whole) have not been deskilled.

Dissatisfied with the deskilling thesis (and its counter-arguments, see Spenner, 1995), Flynn (1988) performed a detailed review of nearly 200 case studies of technological change in the

---

<sup>1</sup> Following Levina and Xin (2007), I exclude from my discussion below those jobs that are exclusively concerned with the production of hardware artifact, its physical installation and the like. For a discussion of software work as the primary task area that defines the focus of this study see section 4.5.

context of one American city (Lowell, Massachusetts). She concluded that a Skill-Training Life Cycle model can best explain the changes in skills, job structures and training opportunities based on the life cycle of technologies. In this model, when technologies are first introduced, tasks associated with them are complex and require firm specific training. Existing jobs are enlarged or new jobs are created to take care of the new technology. As technology spreads and demand for skills grows, new (vocational or university-based) schools emerge to supply skilled individuals. After some time, when technology reaches a level of maturity and tasks become routinized, jobs become increasingly similar and skills transferable. During this period new occupations emerge and take form. When technology becomes stable, job structures become stabilized; schools and colleges develop the required workforce and employers hire individuals to fill specified job positions. When technology becomes obsolete the priorities of educational institutions change and the life-cycle reverts to the introductory phase of a newer technology.

The value of Flynn's approach is that it is the clearest expression of a direct relationship between technological and occupational change. Moreover, for Flynn, changes in IT occupations such as computer programming were among "the classic examples of this transformation" (Flynn, 1988: 18). But as we know, at least since Flynn's analysis has appeared, computer programming has neither declined nor stabilized. Rather, it has become increasingly diversified. The problem with Flynn's approach is that, even though it was developed as part of a program to guide managerial and policy-level decision making about human resources, it takes technological change as a sufficient cause for propelling change through the social environment of work and employment. Indeed, in this view "plant closings, worker displacement, and skill obsolescence are seen as natural consequences of technological progress." (Flynn, 1991: 118).

While the politics of Flynn's approach are very different from those of Braverman (and his followers) they both take a view that over-determines the implications of technological change for occupations. In contrast to these approaches, there are views that take a more contingent approach to occupational change and take a wider variety of factors into consideration. Many of these approaches emerged as part of the literature on the sociology of professions. For instance, based on empirical studies of the medical profession, Friedson (1970, 1984) and Larson (1977) argued that -rather than a natural course of development for specific occupations- professionalization is an attempt by a self-interested social group to transform their skills into an officially recognized set of credentials that could be used as a barrier to entry of others to the labor market. In doing so, they engage with a wide variety of other social actors including clients, the state, universities and bureaucratic organizations.



Following Friedson and Larson's pioneering work, a number of researchers tried to understand changes in IT occupations as attempts to professionalization of IT work. Nevertheless, in their studies, they often found that IT workers have failed to mount such a "collective mobility" project (e.g. Murray and Knights, 1990, Rose, 2002). The main problem that this line of research encounters in studying IT work is that it starts by assuming that an occupation (like IS managers) is a "unified actor" (Larson, 2013: xxv, cf. Fincham, et al. 1994: 275-6). This is despite the fact that, irrespective of the theoretical angle chosen, fragmentation is a theme that often comes up in different studies of IT work (e.g. Kraft and Dubnoff, 1986; Fincham, et al., 1994: 235; Murray and Knights, 1994: 85).

Another group of researchers have drawn on Abbott's influential study of division of expert labor (1988). Abbott envisaged the division of labor among professions as a system in which different occupational groups claim jurisdiction over specific task areas and compete with each other to show that their "somewhat abstract knowledge" is more applicable in that area (p.8). This perspective offers a multi-level understanding of occupations that makes no assumptions about the level at which occupational solidarity may emerge (see also Abbott, 1991b). It also postulates that occupational groups, at different levels, may define, defend and, if necessary, redefine their jurisdictional claims in the light of changes in the environment (including technology, work organizations and the government), the task itself or other occupational groups.

Abbott's analysis is filled with examples from computer occupations and devotes a whole chapter to jurisdictional disputes among what he calls the 'information professions' (consisting of librarians, statisticians, operational researchers, accountants and information systems). Nevertheless, for all its power of exploring inter-occupational competition, Abbott's framework does not provide a way of understanding changes in those categories of computer work that are not often considered theoretical (e.g. maintenance, see Zabusky, 1997). Focusing on the knowledge components of academics (Somers, 2010) or professional associations (Adams, 2007), researchers using Abbott's framework have been unable to account for the trends among the IT workers.

In more recent years, researchers have tried to understand changes in occupations as a consequence of increasing levels of organizational control over work (Abbot, 1991a; Reed, 1996; Leicht and Fennel, 1997). While once there was a sense of alarm that occupations may become increasingly bureaucratized and rigid (e.g. Larson, 1977), the rise of flexible forms of organization and new managerial strategies that grant greater autonomy to workers have

led some social theorists to suggest that occupational boundaries are increasingly disappearing (Casey, 1996; Smith, 1997; Kallinikos, 2003). Drawing on this literature, Damarin (2006: 451) found that website production is characterized by a “modular occupational structure” in which “differentiated work roles are combined in configurations that vary across different projects and organizations, yielding fluid job definitions and “portfolio” careers”. In her view, this finding neither confirms nor rejects the view that occupational boundaries are disappearing: it merely points that the commonly held views about occupations as horizontally distinct work domains may not be accurate.

But Damarin’s findings (as she admits) are far from conclusive and leave much to be desired. Her research design is based on interviewing and examining the resumes of 60 website developers in New York over a period of 3 years, supplemented by an unspecified number of job ads. As she explains, her approach to the research changed when she found that many jobs in participants’ resumes did not fit into any of the “occupational boxes” that she had identified in her preliminary research (p. 439). This led her to take a grounded theory approach to understanding job categories. While this is unproblematic as far as the examined group of workers is concerned, it is unclear how generalizable her findings are for the wider IT occupations. Damarin does not provide a background about the emergence of website production as a category of work (presumably due to lack of space, see also Damarin, 2013). More importantly, she does not incorporate work histories of participants before entering website production into analysis. It appears that, in her analysis, no institutional background could have had an impact on the work practices that she had observed. (She only provides gender and degree levels attained by participants at an aggregate level). In other words, there is no historical or external context against which the identified structure can be put into perspective. Therefore, she fails to explain why the structure of website production is modular and how it has come about.

In sum, various social theories that have been used to explain the changes in IT occupations have failed to grasp the multiplicity of aspects that affect it. They either over-determine the context of work and the forces behind it (Braverman and Flynn) or make assumptions about it that do not necessarily hold (Larson about occupational solidarity, Abbott about theoretical component). They also usually either focus on one or another job category (Kraft’s programmers/coders, Knight and Murray’s IS managers, Damarin’s web developers, etc.) without considering other jobs, or focus on a specific aspect of IT work like its knowledge-base without considering work at all. The result is that those who have tried to understand the historical developments of this field have been left to their own tools. Since I incorporate a large part of this literature in my historical narrative, I will not review them

here. The only exception is the body of work produced by Friedman and Cornford (Friedman and Cornford, 1989; Friedman, 1990), which to my knowledge, is the only systematic (model-based) historical study of IT occupations. In the next section, I consider this model, the insights it has produced and its limitations.

#### 2.4. Occupational change in the context of IS departments

Based on a detailed historical study of changes in IS departments in the U.S. and U.K., Friedman and Cornford's work (1989) provided a conceptual model and a phased history that would explain why concerns, practices and structures of IS departments changed over the period between early 1950s and early 1980s. Since the details of their phased narrative cannot be summarized here, it suffices to consider the 'static' model that underlies their narrative (a summary of their argument can be found in Friedman, 1990; I will compare my findings to theirs in the empirical section). They identified computer workers by their mediating role between the computer itself and its users in the organization through development of 'uses' (i.e. applications). In Friedman and Cornford's view, computer workers also mediate between users and external suppliers of technology (usually hardware). Based on this definition, Friedman and Cornford identified four external 'agents of change' in the evolution IS departments: computer technology (changes in software and hardware), uses of technology, conditions of the labor market and competition from external suppliers of the same services. They argued that technical practices, task structure, employment relationships and managerial control strategies in IS departments have changed in relation to these sources of change.

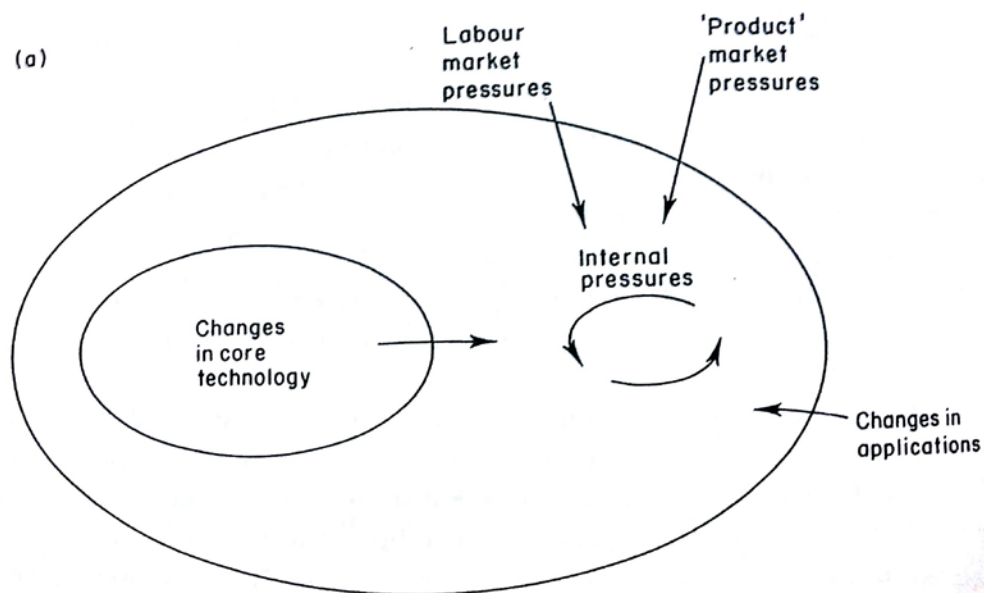


Figure 2-1: Friedman and Cornfield's (1989) model ('static picture') of changes in computer systems development

Using this perspective, Friedman (1990) makes several observations about changes in IT occupations:

- 1) IT occupations, especially programmers, were neither eliminated nor deskilled (as Kraft and others had predicted).
- 2) IT workers continued to flourish mainly due to the expansion of uses of computers across organizations. Nevertheless, there seemed to be a constant shortage of them.
- 3) The new applications and technologies changed the kinds of skills that were required for computer occupations.
- 4) Correspondingly, over time, people from a wider variety of backgrounds and skills entered the labor market.

Based on these observations Friedman predicted that there would be more emphasis on analysis skills and also a rise in the share of industry-specific applications and therefore, demand for industry-specific skills. He also predicted that growing number of job ladders and career structures will be developed within organizations that would “tie computer people to employing organizations” (p. 794).

As far as it is visible within user organizations, Friedman’s analysis goes a long way in showing how computer occupations have become differentiated. In doing so, it also shows how technological change and organizational and managerial flexibility, each can play a role in the dynamics of computer jobs. Nevertheless, his predictions, in retrospective, expose the limits of his approach. Perhaps his most conspicuous error is that, since his predictions were made, computer workers have become increasingly detached from (rather than attached to) their organizations. Not only many of them frequently cross the boundaries of organizations over the course of their career, but also an increasing number of them work as freelancers (Barley and Kunda, 2004; Osnowitz, 2010; Joseph, et al., 2012). Friedman also takes for granted the continual availability of adequately skilled individuals and an almost barrier-less labor market. In other words, by focusing exclusively on hiring decisions from the perspective of IS managers, he overlooks wider developments in the labor market including the provision of training and norms of employment relationships. While this is appropriate for a study that focuses on the internal dynamics of IS departments, a study of IT occupations must take into account this wider institutional framework.

## **2.5. Summary and conclusion**

In this chapter, I have reviewed various theories of occupational change and how they have been applied to IT work. In every case, as I have shown, they have failed to capture some essential aspect of IT work that, with the passage of time, has proved detrimental to their analysis. Before closing this chapter, I must deal with another possibility that, as far as I

know, has never been developed and put to test, but has always been around as a pseudo-explanation. Maybe it is the very nature of IT that enables continuous (and disruptive) changes in IT work and therefore reshapes IT occupations. For example, one of the findings of Damarin (2006: 455) was that the modular structure of web production “rests on fluid, role-spanning jobs and careers, often oriented to the Web as a whole rather than to particular specialties or employers”.

IT workers have long been identified by their association with the tools they command; those tools are, to use Abbott’s terms, the foundations of their jurisdiction. Commenting on the objective vs. subjective qualities of tasks of professions, Abbott (1988: 39) observes:

“A profession is always vulnerable to changes in the objective character of its central tasks. Thus, in the immediate postwar period, computer professionals were generally electrical engineers or programming specialists expert in hardware peculiarities of particular machines. The development of compilers and other forms of fixed software has freed computing professionals to develop rapidly towards the much more open applied software jurisdiction. The development of UNIX and other hardware-impervious operating systems will allow complete emancipation.”

The emancipation that Abbott predicted could be as equally a cause for unification as it is now, at least apparently, a cause of fragmentation. Ever since programming was separated from the specifics of hardware, the success of programmers in defining and defending their domain has depended on the degree to which software is “amenable to cultural work” (Abbott, 1988: 36).

This, however, does not mean that software practitioners have had to either find a “silver bullet” (Brooks, 1987) or, for the lack of a better expression, convince others that their bullet is made of silver.<sup>2</sup> Neither software, nor software practitioners have been constant objects. As Kallinikos (2004: 141) has argued, the so-called recalcitrance of IT artifact at any point is itself a product of “the historical conditions under which particular technologies emerge and develop, and the forms by which they have become institutionally and socially embedded”. Thus, changes in IT-related tasks must be understood along the historical context of emergence and profusion of IT occupations and vice versa (see section 4.5.1). In the next chapter, I provide a theoretical framework for understanding historical development of IT occupations in their institutional context that takes into account both the object of their work as well as the changing role of IT workers.

---

<sup>2</sup> Abbott’s view of computer professionals assumes too readily that their main task is defined around the technological object. Even though this is debatable, I am not questioning it here. After all, the main problem is that we do not know what specifically these jobs involve. For a view of ‘software crisis’ as social construction, see Ensmenger (2001b).

## 3 Definitions and Theoretical Framework

### 3.1. Introduction

My primary question in this research is why IT jobs have not been institutionalized. In the previous chapter I have shown that existing theories do not provide an adequate framework for understanding IT work. In this chapter I will draw on the existing studies of political economy of skill formation and employment systems to construct a generic theoretical framework for understanding occupations. This will result in identifying the main actors in the dynamics of formation of occupations. Then I will apply this model to the particular features of the American IT industry (my case study) to develop an explanatory framework for understanding the changes in IT jobs in that particular context. But before building my theoretical framework, I must provide some basic definitions, especially about the notion of ‘job’ and what is meant by its institutionalization.

A job is a specific set of tasks to be performed by an individual for which he or she expects to receive a payment (Kalleberg and Berg, 1987: 36). In contemporary society, holding a job often involves being entitled to a set of rights and responsibilities that are defined by an employment relationship. Moreover, there are certain structures that bring regularities to jobs and shape the working environment of individuals. Industries, work organizations and occupations are typical structures that can be found in any specific area of work. These ‘work structures’ are:

“[T]he rules on which many people have agreed, and thus legitimized, for longer or shorter periods of time, as effective means of...production and distribution. [They] represent the hierarchical orders and clusters of interests, configurations of norms, and the rights and obligations that characterize the relationships among the different types of actors in the economy. These structures describe the ways in which labor is divided, tasks allocated and authority distributed.’ (Kalleberg and Berg 1987: 2)

As such, work structures can be understood as institutions shaping jobs. Institutions can be defined as sets of rules and practices that are either enforced or taken-for-granted because there is an actor (or group of actors) that has a vested interest in them based on some idea (e.g. a means-ends functionality).<sup>1</sup> They are, however, not necessarily designed. They may emerge from interactions between different groups of people and are always shaped by those interactions as well as changes in other institutions. Therefore, the origins of institution and its architects may have been different from those benefiting from it or the way they benefit from it (see Pierson, 2000; Hall, 2010).

---

<sup>1</sup> In this research, I have assumed that there is a degree of convergence between different schools of institutionalism that I draw on (see section 3.4).

<b>Work Structure</b>	<b>Definition</b>
Industry	A collection of profit-oriented work organizations (companies) that provide and sell a group of possibly heterogeneous products that potential buyers see as close substitutes for each other and therefore are traded in the same market (see below). Companies in the same industry may form business associations to coordinate their collective interests.
Work Organizations	Typically formal hierarchies that are designed to facilitate the production and distribution of products and services by more or less well defined mechanisms of control and collaboration between different organizational roles. They can be public or private, part of one or more industries and employing workers from a variety of occupations. Unlike industries and majority of occupations, they enjoy well-defined legal status and can claim rights and liabilities like individuals.
Occupations	Aggregations of jobs that perform similar tasks in different work organizations. The more coherent an occupation, the more similar the organizational roles and employment relationships of jobholders in that occupation. If the jobholders in any occupation form collectives to pursue their common interests, their associations may achieve formal representation and official recognition.
Labor Markets	Norms and practices that govern the sale and purchase of workers' skills and capabilities in return for (material or non-material) rewards.
Product Markets	Markets in which products and services are bought and sold.

Table 3-1: Work structures (adapted from Kalleberg and Berg, 1987)

Occupations are one of the institutions that influence the emergence and shaping of jobs (table 3-1). However, they are also shaped by other institutions. Abbott (2005: 322) argues that, in order to understand occupations as “social entities with coherence and consequence”, any study of occupations must consider how ‘an enduring group of people’, ‘a set of institutions’ and ‘a task area’ are brought together:

“Any serious study of occupations must begin with the question of how and when these three strands of occupation-ness can be brought together. More important, it must also understand how lineages—consistent social structures through time—can be created within each strand. How can or does a task area remain unified across time? How does a group of people maintain a position in the division of labor as they and the institutions around them age?”

As I mentioned in the previous chapter, it is increasingly recognized that occupations are changing shape under the influence of organizations. In the rest of this chapter, I focus on the relationship between the recruiting practices of organizations and the formation of occupations and the central role that attainment of skills play in this dynamic.

### 3.2. Occupations and skills in the age of organizational dominance

Abbott (1989) has suggested that the main mechanism through which organizations affect occupations is their recruiting practices. Which individuals they select to fill organizational positions and how they select them have direct effects on who enters an occupation (in the formal sense) and can become a member of the occupation (in the real sense). Abbott also points out that in contemporary society, organizations have based their recruitment mechanisms on individuals' attainment of knowledge and skills, which can be gained through either apprenticeship (in blue-collar work) and advanced education (in the professions).

Far from being a 'natural' process, the emergence of these mechanisms in any area of work is a contended process in which different social groups actively participate. During these processes four aspects of skills are decided:

- 1- What are the required skills?
- 2- Who does have or is best placed to earn those skills?
- 3- Who shall provide (or pay for the provision of) those skills and how?
- 4- What does earning those skills mean in terms of the rewards individuals receive?

For example, in the case of blue-collar work, sociologists and political economists investigating the relationship between labor and businesses in capitalist systems have demonstrated that there are differences between the ways in which vocational training in different countries is institutionalized. Reviewing this literature, Thelen (2004: 20) concludes that cross-national differences in 'regimes of skill formation':

“can be traced back to differences in the political settlement achieved between independent artisans, skilled industrial workers, and employers in skill-intensive industries in the early industrial period. The kinds of settlements that were possible in individual countries were heavily mediated by state action (or inaction) which frequently tipped the balance [...]”. (See also Thelen 2002, 2008)

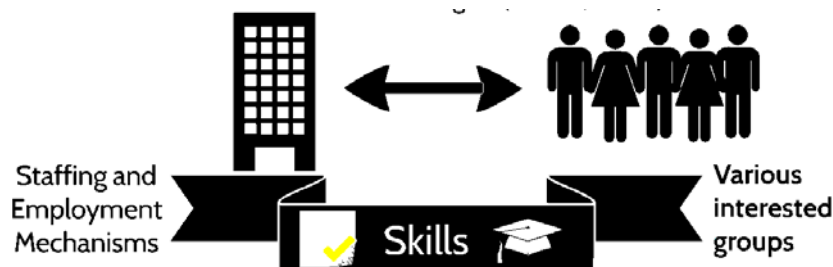


Figure 3-1: The relationship between various social groups who have an interest in taking over a task area and organizations that employ them is mediated by skills.



In the case of professions, we have already seen that they achieved this by gaining autonomy regarding their skills and monopolizing the market while organizing themselves as alternatives to organizations (Larson, 1977, Friedson, 2001; Krause, 1982). However, even in the process of professionalization, the essential components of their contentions and especially their reliance on state have been broadly similar to other skilled-occupations (Collins, 1979; Rueschemeyer, 1983; Siegrist, 2002). In any case, in recent decades professions have increasingly joined the ranks of employees, becoming a part of what is commonly known as knowledge workers (Crompton, 1990; Blackler, Reed and Whitaker, 1993; Brint, 1994; Evetts, 2011). The same is true for the wider categories of white-collar work that emerged within large corporations (Mills, 1951; Strauss, 1963; Rueschemeyer, 1986).

Moreover, as Fligstein (2001) has argued, the settlement about regimes of skill formation is achieved at the same time as the settlements about other aspects of work. Fligstein distinguishes between three employment systems that have emerged since the industrial age. They include vocationalism, professionalism and managerialism. Managerialism is distinct from the other two in that, under managerialism, managers have the ultimate say in deciding who enters a job and they often select from among university graduates.<sup>2</sup> Even though Fligstein’s characterization of these employment systems is debatable, it is clear that -as ideal types- they demarcate three different sets of agreements about how workers are organized, how the skills they need to be employed is transferred to them and how they make progress (or move) from one job to another.

<i>Attribute</i>	<b>Vocationalism</b>	<b>Professionalism</b>	<b>Managerialism</b>
<b>Affiliative Unit</b>	Trade union	Professional Association	Firm (or company union)
<b>Training</b>	“On the job”	Postgraduate Education	University
<b>Career</b>	Intra-occupational	Intra-occupational	Intra-firm

Table 3-2. Three types of employment systems according to Fligstein (2001: 105). It is notable that IT work does not fit into any of the three categories.

Both Fligstein and Thelen agree that the main actors in the process of reaching a settlement about the regime of skill formation and employment relationships are workers, employers and the state. Moreover, in both analyses, the way that different actors align with each other is the most important factor in shaping the outcome. While neither Fligstein nor Thelen consider occupations as central elements in their analysis, it is clear that for them, formation

<sup>2</sup> It seems that Fligstein has in mind (white collar) employees of corporations as the prime example of managerialism. He cites Edwards (1979) and Burawoy (1985).

of enduring groups and collective action is a prerequisite for participation of workers in the dynamics of employment systems and skill formation and they often do so by forming occupational groups (see also Nelsen and Barley, 1997). Therefore, one can argue:

- 1- Employers influence the formation of occupations through their staffing decisions. In contemporary society, these decisions often have implications for defining the regime of skill formation and, at the same time, the shape of employment relationships in the labor market.
- 2- The interactions among employers (work organizations) and workers over formation of skills and shaping of employment relationships is a political (i.e. power-induced) process in which distribution of alignments between different actors can shape the outcomes. State plays a crucial role in mediating this relationship. Other organizations like those providing training can also play a role.

The role of state in this process requires some clarification. A basic character of capitalist societies is that the state has a limited direct role in the administration of economy. Nevertheless, it plays an important in shaping the economy and, in particular, the labor market. Therefore, worker associations (like unions and professions) and companies are likely to lobby with the government to promote their interests. More importantly, through setting up broad “regulatory regimes” that provide the framework for employer-employee relationship, the state can influence the framing of debates (how different groups articulate their interests), and the likely alliances. Above all, regimes of skill formation are often established through national level training and education policies of the government. It also provides mechanisms for rule of law and resolution of conflicts through the judiciary. Therefore, even if it is not a completely unified body, because of its special position in relation to other groups, it can be considered a single actor (Rueschemeyer, 1986; Kalleberg and Berg, 1987; Thelen, 2004).

It must be noted that, the historically liberal political environment of the United States has precluded the emergence of centralized training regime or highly regulated employment systems (Thelen, 2004, Fligstein, 2001: 113-4). Moreover, particularly in the context of American IT industry, which is the subject of this research, IT workers have not been politically very active (either as unions or professional associations) in the shaping of regimes of skill formation or employment systems. But this does not affect the applicability of theoretical framework described above for two reasons. Firstly, under such circumstances, formation of active collectives of employees can be traced at company or even lower levels. The task of the researcher is to see if those collectives have led to any short term or long-term results at those levels.

Secondly, the literature in political economy of business and labor cited above does not assume that employers and workers are necessarily adversaries. Their historical studies have shown the possibility of alignments between diverse (and potentially, adversary) actors (see Thelen, 2002, cf. Fligstein, 2001: 56). One stream of research in this literature draws attention to what it calls ‘cross-class coalitions’, that is, alliances between employers and workers (or capitalists and labor) in workplaces. One form of these alliances occurs in the context of conflict of interests among employers. From this perspective, attention must be focused on competition between different employers and, especially, the decisions of dominant employers (Swenson, 2002). The more dominant a company is in competition, the more influential it is likely to be in shaping the regime of skill formation and employment system. This is not only because the dominant company is more likely to be ‘heard’ or imitated (by the state or competitors, respectively), but also because they can use their position in the product market to impose their favored arrangement in the labor market and vice versa. Because this norm is not suited to the priorities of other companies, whether or not they adapt it they are likely to incur costs and this would strengthen the position of the dominant company. Thus, competition -as a factor determining relative support for alternative institutional arrangements- is a central element in the narrative and analysis of struggles between actors (mainly companies) in this research. (Correspondingly, government’s role in regulating the competition has also to be considered.)

But competition to in the IT industry does not have an obvious relationship to developments in the regimes of skill formation or employment systems in IT work. In the political economist approach cited above, on the other hand, while competition is considered as a relevant factor in the institutional environment of emergence of regimes of skill formation, technology (and technological change) is largely treated as an exogenous factor. In her detailed review of historical studies of industries and worker-employer relations, Thelen (2004) frequently refers to technological change and competition between employers in the product market, but does not relate the two to each other.<sup>3</sup> This is a limitation for our purposes because, as a tool and as a product, IT is intimately involved in the changes of IT work and competition in the IT industry. Therefore, in order to understand how competition has affected the shaping of regimes of skill formation in IT work, it is imperative to establish a theoretical link between technological innovation and the shaping of regimes of skill formation.

---

<sup>3</sup> This link is also ignored in Fligstein’s approach.

### **3.3. Linking regimes of skill formation to competition in the IT industry**

Institutionalist research in organizational and IS studies has convincingly shown that in order to appeal to other actors (companies, workers, customers, etc.), innovation in products must be relevant to, among other things, a common vision and a shared pool of skills (Aldrich and Fiol, 1994; Swanson and Ramiller, 1997, 2004; Lanzara and Patriotta, 2007; Wang and Swanson, 2007; Wang and Ramiller, 2009; Sine and David, 2010; Scott, 2010). Technology-based competition, therefore, is only possible when within -as well as outside-organizations, a pool of skilled workers exists for exploring and trying out multiple combinations of old and new organizational arrangements and technological artifacts (Tiwana, Konsynski and Bush, 2010; cf. Gawer and Cusumano, 2002: 5, 62).

At the level of organizations, this means that companies may compete by drawing on and combining the resources (especially skilled workers) that are available internally and externally (see Ciborra, 1996). Indeed, when a company dominates the product market as well as the labor market (as IBM did through the 60s and 70s) instead of ‘platform competition’, its strategy can be based on providing “total-systems” (see Campbell-Kelly and Aspray, 1996: 131-2). The results of these decisions and actions, in turn, shape the shared pool of skilled workers (see also Scarbrough, 1995; Mueller and Dyerson, 1999). Therefore, by influencing the available pool of skilled workers and how it is deployed, technological and organizational decisions of companies competing in the IT industry can affect the labor market. This is especially true about the dominant employers who can directly affect large sections of the labor market.

Decisions of the dominant companies are also influential in the regimes of skill formation in a more fundamental sense. They can use their power in the product market to increase or decrease the value of certain skills. Given the increasing vertical disintegration of industry (Bresnahan and Malerba, 1999; Bresnahan and Greenstein, 1999; Baldwin and Woodard, 2009) and importance of technological innovation that relies on the existing layers of technology (Yoo, Henfridsson and Lyytinen, 2010), powerful companies in the industry often have the option to block or adopt other technological products on their platform. For example, if a dominant company refuses to adopt a technology on their platform, depending on the success of that strategy, the skills associated with using or developing that technology may lose part of their value because they lose potential customers. On the other hand, when they embrace a product on their technological platforms, they boost the potential demands for skills associated with that product. For instance, IBM’s decision to use Linux has boosted the value of Linux skills since the early 2000s. Therefore, there is a possibility of direct link between product decisions of companies and formation of skills in the industry.

Since decisions regarding the use of internal and external pool of skilled workers are often based on the business model of company, one must pay attention the platform of IT work and its value contribution. In other words, how internal employees and external skilled workers complement the success of technological innovations (e.g. Popp, 2011) has important implications for how companies try to deploy them. In most cases, company's business model is the basis of company employment policies and shapes its preferences in terms of the preferred arrangement for formation of skills in the industry. This means that over time, as the production of digital products and services becomes more and more diversified and layered (Steinmueller, 1996), and simplest operations (like clicking on a hyperlink) draw on the technologies of several companies, every company has to make decisions about how it is going to draw on the available human resources (internally and externally) depending on its position in the structure of industry.<sup>4</sup> It may also need to invest in promoting skills associated with their adopted business model and technological products (Baldwin and Clark, 2000; Yoo, Henfridsson and Lyytinen, 2010, Ghazawneh and Henfridsson, 2012).

From a managerial standpoint, these decisions can be about one of the following:

- 1) Employment policies and practices and hiring decisions
- 2) Training practices and certification programs (internally and externally)
- 3) Reorganization efforts such as spin-off, restructuring, merger and acquisition

Among these, employment practices and training are the most directly relevant issues because they concern the availability of skilled workers in the labor market (see Mueller and Dyerson, 1999). Employment practices and decisions include those decisions that affect hiring criteria, employment security<sup>5</sup> policies and the use of temporary or part-time workers. I will also focus on layoffs (i.e. decisions that result in mass job loss) because, arguably, rather than any objective (de)valuation of skills being the cause, it is these decisions that give value to or devalue skills of individuals (see Kaplan and Lerouge, 2007; Joseph, Ng, Koh and Ang, 2007; Goles, Hawk and Kaiser, 2008; Zhang, Ryan, Prybutok, Kappelman, 2012). Moreover, as causes for grievance, layoffs are often the most common base for formation of worker collectivities.

---

<sup>4</sup> Vertical disintegration can be triggered by technological innovation, tactics and strategies of competing companies, or regulatory frameworks and legal decisions (see Funk, 2012).

<sup>5</sup> Employment security can be defined as "a practice that protects full-time workers against loss of employment and earnings for reasons unrelated to their job performance or behavior" (Loseby, 1992: 9). Under special circumstances it can be extended to part-time workers but not temporary workers. It does not imply guarantees about a specific job but rather continual employment opportunity for an employee.

Training programs can have a number of roles: internal trainings increases the pool of competent resources in the company while external training programs for (potential) customers or other workers often boosts the market for company's products. In both cases, training programs help to establish what the relevant skills are in the relevant areas of work. If successful, a *de facto* dominator in the market can define the set of required skills and license developers for that market (e.g. Microsoft developer certifications; see also Nardie, 2011: xi). When certain skills become commonplace, it is usually a sign that the products of a company have become dominant in the market (e.g. Microsoft products and ICDL programs).

Reorganization efforts have more complicated effects. When skills (or expertise) are the main concern, mergers and acquisitions could be directed at having access to the expertise of another organization or the skills of certain individuals in that organization. Restructuring and spin-offs, on the other hand, are usually undertaken for legal, ownership or financial purposes. They may also be concerned with the potential risks to profitability or productivity. In any case, they affect how a company organizes its skilled workers and can affect the shaping of skill formation. Similarly, in spin-outs - that is when an employee or a group of employees leave to form a new independent firm- they alter the balance of skills available to the company and in the market. On the one hand, they build on skills acquired in the former company and, on the other hand, they often bring into play a new skill-set that was previously inexistent or ignored. As we will see, this type of spin-outs are very frequent in the history of IT industry and more often than not are the result of former company's unwillingness to adopt and nurture those skills (and their embodiments, i.e. new products).

To sum up, in this section I have argued that within a context of technological innovation, skilled workers play a central role in the shaping of competition. In particular, the way in which technology and skilled workers complement each other in the business model of companies is central to their position regarding what skills are to be invested in and how (that is, under what regime of skill formation). Because of their power in the industry, decisions and policies of dominant companies –especially regarding their employment practices, training programs and reorganization efforts- play a significant role in shaping the dynamics of skill formation. Therefore we can seek to understand how the decisions of dominant companies have shaped the labor market and how other companies and workers have responded to them (by adopting, challenging or adapting to them). The interactions between these actors can help us understand how job institutionalization (or lack thereof) is linked to corporate decisions within the context of competition in the industry and the broader socio-political environment.

### **3.4. Placing the theoretical framework within the institutional theory**

Having put forward my theoretical framework, I must now place it within the literature on institutional theory. It must be noted that, even though institutional analysis has a long history in the social sciences, interest in theorizing about institutions and their role in the society has been on the rise since the 1970s. These “new institutionalisms” are spread across the various disciplines of social science and do not form an integrated or even coherent theoretical body (Peters, 1999; Scott, 2014). What different varieties of institutionalism (old and new) share is the view that “current actors and events are greatly shaped by past efforts and their enduring products” (Scott, 2014: 1; compare to Peters, 1999: 18). But beyond this common interest, there are important conceptual and methodological differences among approaches to institutionalism and even a common definition of “institution” has remained elusive. For the purpose of this study, I focus on the theoretical approaches that concern the study of occupations and inform my understanding of institutions (for a review of developments in organizational institutionalism, see Greenwood, Oliver, Sahlin and Suddaby, 2008).

Until the 1970s, sociology of occupations was dominated by a certain variety of institutionalism that was rooted in the Chicago School (Abbott, 1992a). Everett Hughes, one of the most influential sociologists of this school, defined institutions as “a collective enterprise, carried on in a somewhat established and expected way” in which people are expected to “take their places” (e.g. families, factories and schools, see Hughes, 1942). For him, occupations were similarly, if less obviously, organized structures (as in unions and associations) but sociologists’ focus should be on how these structures are created rather than whether or not they exist in any specific case. Under his influence, researchers focused on the “interactional” basis of institutions, i.e. the role of interacting individuals in enacting and prolonging institutions (see Barley, 2008).

This was combined with another tradition of the Chicago School, namely, the idea that occupations have “a natural history”; that they follow through a relatively predictable course that leads to establishment of those structures. In the case of study of professions, this resulted in several theories of professionalization process as a sequence of definite and progressive stages. In these histories, interacting individuals and groups bring into force institutional forms that lead to the establishment of the profession. For instance, Wilensky (1964) identified the following stages: founding a training school and later a university schools, forming local and national associations, passing of state-level licensing laws, and finally, establishing a code of ethics for practitioners (see Abbott, 1988, 1991b). From this

perspective, many occupations with professional ambitions had simply not “fully traversed” the process of becoming a profession (Abbott, 2008).

The contemporary institutionalist view on professions within organizational studies, as articulated by Scott (2008), follows in the same steps in that it views ‘professionals’ as institutional agents (see also Leicht and Fennel, 2008). But while most studies of the Chicago School tradition focused on cognitive and normative aspects of professions (Friedson (1986) being a notable exception), now professionals are seen as actors that enact the cognitive, normative *and* regulative frameworks of professional work (e.g. theoretical/scientific outlook, code of ethics, licensure, etc.). Indeed for Leicht (2005: 604) professions are “macro-level institutions” that embody “distinct and identifiable structures of knowledge, expertise, work, and labor markets, with distinct norms, practices, ideologies and organizational forms”.

As novel and productive this theoretical angle may be, it inevitably falls into the same definitional quagmire that has troubled sociology of professions for a long time, i.e. what ‘exactly’ are the distinguishing features of professions (see Evetts, 2011). In practice, sociologists working within this perspective seem to focus on well-known professions such as medicine (e.g. Scott, et al, 2000) even though they occasionally view relatively obscure occupations as professions as well (such as museum curators in DiMaggio, 1991).<sup>6</sup>

More importantly, this particular perspective on ‘professions as institutions’ cannot provide the analytical tools for understanding occupations in transition, i.e. how occupations may come (or fail) to be established as a profession. Even in the case of recognized professions, scholars taking this perspective consider elements such as market and technology as elements in the “institutional and technical environment” of professions and do not provide a systematic theory for understanding how changes in this ‘environment’ affect professions (see Leicht and Fennel, 2001, 2008).

---

<sup>6</sup> A distinct approach in institutional theory, namely institutional logic, treats professions as an ideal typical institutional order, that is, “a governance system that provides a frame of reference that preconditions actors’ sense-making choices” (Thornton, Ocasio and Lounsbury, 2012: 54). Building on Thornton’s (2004) study of changing conditions of work of book editors, whom Thornton takes as “professionals”, professions are distinguished as taking the “root metaphor” of “relational networks”: relations between individuals whose legitimacy is based on “personal expertise” and whose source of identity is “association with quality of craft” or “personal reputation” (Thornton et al., 2012: 56). Such a micro-level conception of professions runs contrary to the history of most well-recognized professions as well as much of the sociology of professions. No wonder there is little (if any) discussion of training or skills in Thornton (2004) outside the ‘new’ context of managerial corporatism.



Most crucially, what this literature ignores is that professionalization is a contested process in which different social groups and actors play different roles using a variety of resources including technology. The literature on occupational control (reviewed in the previous chapter) brings this aspect of work to the fore and tries to focus on what different mechanisms occupations may use in order to gain control over their work. In this context, the work of Abbott (1988) is of particular importance because it explicitly tried to define professions as occupations employing a certain mode of competition (i.e. knowledge-based struggles over jurisdiction) rather than having specific internal characteristics (see also Gorman and Sandefur, 2011). Moreover, he understood professionalism as “the main way of institutionalizing expertise in industrial countries” (Abbott, 1988: 323). He argued that professionalism exists, in part, because (p. 324):

“[M]arket-based occupational structure favors employment based on personally held resources, whether of knowledge or of wealth. Such employment is more secure, more autonomous, and usually more enumerative, partly because it is organized in [professional] careers...[which offer] continuous independent life chances”.

In this formulation, professionalism exists to provide a certain function and, in doing so, relies on the wider institutional context (including market and work organizations). But it is not only professional jobs that rely on this wider context; all jobs are shaped by this wider context which Kalleberg and Berg (1987) have called “work structures” and as I argued before can be understood as institutions.

Thus, the objects of institutional analysis in this study are the various work structures that I have listed in table 3.1. As I have mentioned before, they all represent sets of rules and practices that are either enforced or taken-for-granted because there is an actor (or group of actors) that has a vested interest in them based on some idea like a means-ends functionality. What is distinctive about this conceptualization of institutions is that it differentiates between actors that are involved in the shaping of institutions and institutions themselves (cf. Scott, 2014: 56).<sup>7</sup> Thus, I consider professional associations, companies and even the state as actors while considering occupations, industry, work organizations, labor markets and product markets as institutions. This distinction is arbitrary but not without ground; it follows the focal point of interest in the research, i.e. explaining the developments in IT work in terms of decisions of companies and the state. One may take any one of actors in my history (like the state, professional associations, or the state) as institutions in their own right and open them to institutional analysis. My aim however is to understand the context (work

---

<sup>7</sup> Actors are not necessarily members of the institution that they are shaping.

structures/ institutional arrangement) of formation of IT jobs and therefore, I take those entities as actors.

Moreover, this approach is consistent with the institutionalist approaches that inform this study. Several authors have noted the complementarity of new institutionalism in organizational studies and economic sociology and historical institutionalism in political science (see, Hall and Taylor, 1996; Peters, 1999; Nee, 2005; Werle, 2012). While their conceptions of institutions do not exactly match, they have enough similarities to provide a backbone for this study (see table 3.3). My theoretical framework builds on both approaches by focusing on the interactions between actors identified in either approach in shaping the institutional context of formation of IT occupations (labor markets, product markets, industry, etc.).

	<b>New Institutionalist Organizational studies</b>	<b>New Institutionalist Economic Sociology</b>	<b>Historical Institutionalism (in analysis of skills formation and innovation)</b>
<b>Behavioral assumption</b>	Actors' decisions and actions are shaped by the cultural belief of their institutional environment	Actors are motivated by interests which are usually shaped by shared beliefs, norms, and network ties	Actors' pursuit of their interests and their responses to institutional context are historically shaped
<b>Definition of institutions</b>	Rationalized myths, logics and routines that endow legitimacy	Interrelated system of institutional elements—informal and formal—facilitating, motivating, and governing social and economic action	Building-blocks of social order representing socially sanctioned (collectively enforced) expectations with respect to the behavior of specific actors or to the performance of specific activities.
<b>Actors</b>	Professionals and organizations	Organizations (including the state) - Individuals articulate interests within organizations and networks	Collectivities, organizations and the state
<b>Macro-level Mechanisms</b>	Isomorphism	State regulation, market mechanism, collective action	State regulation, collective action

Table 3.3 : Varieties of new institutionalism, based on Nee (2005), Hall and Taylor (1996), Streek and Thelen (2005) and Werle (2012).

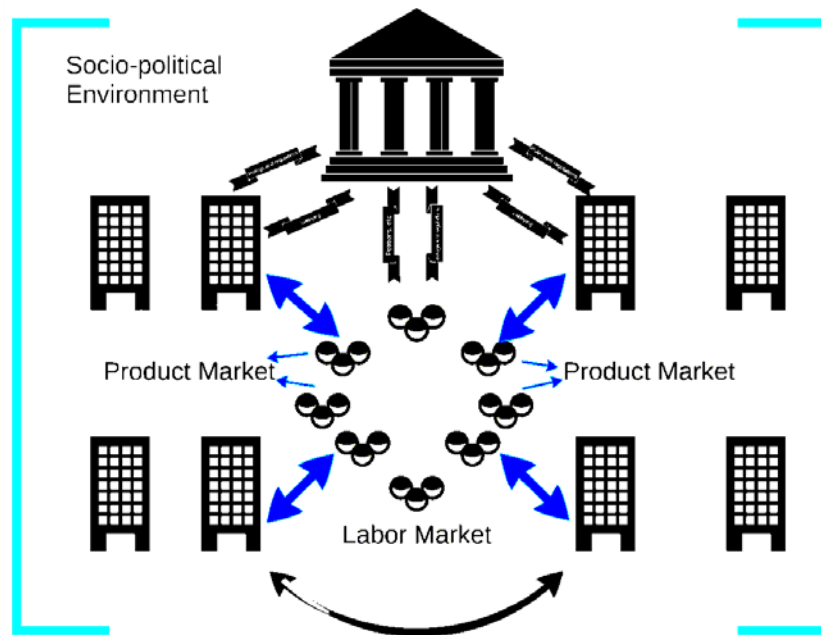


Figure 3-2: Diagram representing the relationships in the theoretical model.

### 3.5. Conclusion

In this chapter, I have argued for studying the formation of IT occupations using a historical approach that concentrates on the interactions of multiple actors which together shape the institutional context of IT work. The key actors in this framework are companies (especially, dominant corporations), worker collectives and the state. The power-induced interactions that simultaneously shape regimes of skill formation and employment relationships occur between 1) different employers 2) employers and employees (or labor market) 3) employers and the state and 4) employees and the state. The first set of interactions occurs mainly through competition in the product market. The second set of interactions consists of employment decisions and policies of companies and the reactions of workers at both company and industry level. The third and fourth sets of interactions occur mainly through lobbying and regulation.

I have further argued that developments between actors must be traced in different domains, including: socio-political environment, market structure, labor market and IT delivery practices. The socio-political environment is the domain of interaction between employers and the state and employees and the state. It can be affected by many other elements that are beyond the control of any specific actor (e.g. war). The market structure is the domain of interaction between employers (as competing companies). In order to understand changes in this domain, I need to study the unfolding structure of industry (its vertical disintegration) as well as the state of competition between different companies. The labor market is where

interactions between employer(s) and employees occur. Its size, entry conditions, employment relationships as well as training arrangements (how training is provided) are important. Changes in this domain include HR decisions and worker/employee reactions. The final domain, IT delivery practices, is a feature of individual companies. It concerns the role of skilled workers (either as employees or in a different relationship) in the business models of companies, which in turn, typically depends on the position of company in the industry. I have also highlighted that the more powerful companies play a more important role because of both the direct influence of their decisions in the labor and product market and the importance of the support they can provide for alternative regimes of skill formation or employment system.

Elements of the theoretical explanation	
	State of competition
Market Structure	Vertical disintegration
	Decisions of dominant companies
Labor Market	Size (number of workers)
	Entry conditions (de facto)
	Training providers
	Worker groups, associations, etc.
	Employment relationships (norms and practices) and decisions
Sociopolitical Environment	Regulatory framework
	State actions
IT Delivery Practices	Object and platform of IT work
	value contribution of IT work

Table 3-3: These elements will be used to structure the narrative.

It is worth emphasizing that the rules governing the fate of IT jobs and IT workers are unlikely to be the result of design of any of the actors in the study. A glance at the history of IT industry in the U.S. (e.g. Campbell-Kelly 2003, Ensmenger, 2010) shows that corporations and individuals entering IT jobs were far more concerned with their short-term interests to worry about institutional design. Nevertheless, the outcome of their interactions has shaped a work environment that is real and can be felt in the life of individuals. In short, nothing is personal and everything is personal (see Sennet, 1999). My argument, briefly stated, is that in the absence of favorable governmental intervention in support of workers, the strategic and tactical decisions of IT corporations involved in the dynamics of competition in the product market have destabilized IT jobs and effectively prevented the formation of a solid occupational collectivity that would lead to institutionalization of jobs.<sup>8</sup> In the next chapter, I discuss how this may be studied from a methodological point of view.

---

<sup>8</sup> By strategic and tactical I mean decisions that are taken as a policy decision or a one-off decision. When IBM stopped offering employment security this was a strategic decision. When they (at the same time) reduced the number of employees by more than 100,000 this was tactical. I also include in this definition moves such as lobbying etc. that have consequences for the shaping of jobs.

## **4 Methodological Considerations and Notes on Historiography**

### **4.1. Background**

The institutional framework developed in chapter three has an explicitly historical orientation in its formulation of both the research question and research design. In this chapter I explain the methodological underpinnings of my research. After this introduction, in this section, I briefly discuss the shortcomings that historical research is often believed to be prone to. Then I will explain the principles of process tracing that I have used in my research (section 4.2.) and how it shapes my research (section 4.3.). Subsequently, I will discuss the ontological and epistemological basis of my research. In the final section of this chapter, I provide the details of resources I have used and my rationale for using them.

There has been an increasing number of historical studies in the IS field since the mid 1990s, but the total number of historical studies compared to the overall output of the field remains small (Mitev and De Vaujany, 2012, cf. Mason, McKenny and Copeland, 1997). Several reasons may account for this. One is surely the relative youngness of the field. Another is the consistent focus of the field on ‘the new and exciting’ (Land 2010). Still, more important reasons should be sought in the historical shaping of the field. Following the early divide between the positivist and interpretivist camp in IS research, the former used history as background data while the latter focused (almost exclusively) on case studies. Walsham (1993: 14), reputedly the most cited methodological guide in interpretive IS research, saw historical research mainly as ‘supplementary’ to longitudinal ‘in-depth case studies’. Even since the rise of historical studies in IS, most studies focus on a single organization rather than a number of organizations (cf. Mitev and de Vaujany, 2012) and certainly not on occupational groups.

It should be noted that training in historical research is lacking across the fields of management and organization studies too (Zald, 2002). Most courses in ‘research methods’ and regularly used textbooks don’t discuss historical research in any detail (e.g Bryman, 2012, Silverman 2013). The most one can expect from these resources is advice on how to use ‘secondary data’ in support of ‘original’ research. Michael Rowlinson (2004) provides a succinct summary of the issues that might have led to the limited use of ‘business history’ in organizational research. Even though his focus is on company archives, the issues he raises are equally applicable to other sources of historical data. The crux of the matter appears to be a perceived “lack of methodological reflection” in “history in general” (p. 301). He

proceeds to explore several “misconceptions” that may have fostered this view about history in organization studies, including:

- 1) History is simply a repository of facts, i.e. ready-made records of past events in archives.
- 2) Archival material is not systematic or comprehensive.
- 3) Archival material has to be merely collected (rather than generated) to be used as historical data.
- 4) Archival materials provide a certain perspective, especially if they are produced by the organization under study.

In Rowlinson’s view, all these issues lead organizational researchers to be suspicious of using archival material. Most of the rest of this chapter will be dedicated to methodological reflection, but I think it is important to clarify these issues first. History is not a mere repository of facts but the result of active and judicious engagement of the historian with available material the same way that an ethnographer (for instance) deals with the fathomless ocean of data that flows around her at any moment. Both have to focus and select what is relevant or not and what account is reliable or not. For the same reason, neither is systematic or comprehensive beyond the limits of their available sources and their methods. For example, ethnographers in the best circumstances can only jot down notes in the flow of action. The notes are to be expanded later and there is always a possibility that some important details are forgotten or mixed with error. It is true that the ethnographer has the opportunity to go back and ask (if she realizes that she has made a mistake) but this is only possible to a certain extent and certainly not after leaving the field. The fact that archival materials present a certain perspective on the issues can also rarely be a serious concern. If anything ‘source criticism’ is one of the oldest tools of historical research (cf. Howell and Prevenier, 2001). Moreover, each of the interviewees that talk to the organizational researcher presents his or her own perspective and it is ultimately the researcher who selects and organizes (and by implication, discards part of) the expressed perspectives.

Therefore historical methods can be as reliable as other research methods. What is necessary instead is taking proper caution and using systematic criteria in the use of archival material (and other sorts of secondary data). I will discuss these issues with regard to my research in the last section of this chapter where I discuss the sources that I gave used (section 4.6). But before doing so, I will describe the process tracing method that informs my analysis and the way I have used it in sections 4.2 and 4.3 and the ontological and epistemological foundations of my research in section 4.4. In section 4.5, I construct the object of my study and its boundaries in precise terms.

#### **4.2. Process tracing and colligation in historically-oriented social research**

For this research, I draw largely on the guidelines provided by Alexander George and Andrew Bennett on the use of process tracing in social sciences (Bennett and George, 1997; George and Bennet, 2005; Bennett 2010) and Andrew Abbott (1984, 1992) on colligation in historical research. I will show how process tracing can be used in historical research, using examples from my own study throughout. In the next section, I will reflect on the issues raised in this section more systematically to make my explanatory framework more detailed and the assumptions of my research more explicit.

Bennett and George (1997) define process tracing as generating and analyzing data on “the causal mechanisms, or processes, events, actions, expectations, and other intervening variables, that link putative causes to observed effects”. In their view, process tracing -as a method- provides a shared middle ground between historians on one side and social scientists interested in understanding, theorizing and explaining historical processes on the other side (George and Bennet, 2005: 223). It involves examining “diagnostic” pieces of evidence that would support or undermine the explanatory power of alternative theories. Any selected theory is considered a valid explanation only if it can provide correct hypotheses not only about the final outcome but also about the causal unfolding of the processes that led to it. At the heart of process tracing is an acknowledgement of equifinality, that is, the assumption that the final outcome could be the result of separate alternative causal processes (like parallel lines of dominos all of which could cause a final domino to fall).

The argument I am putting forward here is that, in the absence of governmental intervention aimed at stabilizing the labor market, decisions made and policies adopted by IT corporations involved in the dynamics of their product market(s) have led to destabilization of IT workers and effectively preventing the formation of a solid occupational collectivity. My aim is to show that the explanation I provide is supported by historical record. I do not assume that it is the only possible causal process, or even that the process I have traced between these causes and the final effect is the only possible route linking the two.

I began to think about my argument when I was reading across histories of IT industry and IT work, and accounts of changes in the American economy (the rise of ‘new’ economy). Once I built my initial hypothesis that ‘maybe decisions made by corporations have had a role’, I started looking for theoretical foundations and how different elements can be put together. My criteria for choosing (or abandoning) a hypothesis was partly pragmatic: I had

to find an explanation that was broad enough to capture a significant part of the phenomena of interest (changes in IT jobs) and yet did not require special access or impossibly long periods of data gathering.

During the period that I was considering different possible arguments, I started gathering data about companies, IT jobs, IT worker's biographies, etc. In the process, I became increasingly aware of the resources that were available. When I finally decided what line of argument I was going to pursue, I had amassed a large amount of (unprocessed) data (e.g. articles in New York Times about IBM) and identified a number of sources for further data gathering. In all this, I had been unwittingly following the advice provided by George and Bennett (2005). George and Bennett (2005) submit that -at the very least- rigorous process tracing helps to narrow the list of potential causes in the study of any specific phenomenon of interest. To achieve this, researchers should (based on Bennett, 2010):

- 1- Start by gathering clues about what could have caused a certain outcome (based on theory or intuition),
- 2- Select one or a set of causes and make an attempt at gathering systematic and sequential data based on relevant theory,
- 3- Compare the evidence to the predictions of theory: if there are significant differences between expected results and observations either select a different set of causes or make a modification to the theory and try again.

This process is iterated until there is a close match (congruence) between the hypotheses provided by theory and the evidence gathered/produced. The ultimate criteria is whether or not "an uninterrupted causal path" can be established between the putative causes and the final outcome, "at the appropriate level(s) of analysis as specified by the theory being tested" (George and Bennett, 2005: 222).

I don't have a systematic alternative argument that could be followed in parallel to the one that I am putting forward here. However, George and Bennett (2005: 220) recommend using process tracing in single-case studies to disprove claims that a single cause is necessary or sufficient for an outcome. As I indicated in chapter two, my main motivation has been to argue against a technologically determinist explanation. Since nobody has claimed that a purely technological argument can explain the differentiation of IT jobs, my strategy has been based on showing that, in my account, at every level of analysis and stage of the narrative, technological change has not been sufficient for the changes that have followed. In other words, non-technological factors have been equally necessary for later developments.



One consequence of this analytical strategy is that my historical narrative is intervened with analysis. In constructing the narrative and analysis, I have tried to verify my theoretical framework by:

- 1) Looking for data about the actions of different actors,
- 2) Thinking about possible effects of those actions in different institutional domains,
- 3) Searching the available data in those domains for evidence that supports the suspected effects or at least does not contradict them.

In this process, my aim has been to construct a processual account of how events following each other have been causally linked by the actions and decisions of actors that I have identified. But trying to match the empirical narrative to the specific terms of theory raises interpretive issues: deciding whether or not an event has happened or how to describe an event is not always straightforward. It is one thing to say that several computer science schools were established in the 1960s and quite another to say that the programming profession was raising its scientific status. While George and Bennett (2005) do not engage with this problem in any detail, it seems that for them the ultimate resolution depends on the extent to which a historical narrative (in its totality) fits into the category of phenomenon with which the theory is concerned (p. 210-11 and 226). The more the contents of narrative of a study can fit the analytical/empirical focus of theory the more theory is supported by the narrative.

Another interpretive issue concerns the relationship that exists between events that is different from causation, namely, colligation. For instance, when a worker protests to his working conditions, his protest is not normally the cause of protest by other workers; rather, it is a part of the collective protest of workers. It may be the cause, if there is, for instance, a temporal distance between the two events or some other indication. More complex (and longer term) events have more aspects and therefore, are more difficult to analyze. Therefore, when deciding whether or not an event has happened, historians (or historically-oriented social scientists) look for a number of 'indicating occurrences', i.e. those events that can be considered a constituent part (rather than a cause) of another event at a different level (Abbott, 1992b).

Historians (and philosophers of history) have been discussing the problem of colligation (how certain events can be justifiably combined to form an emergent whole) for a long time. (e.g. Walsh, 1958, McCullagh 1978, 2004, Roberts 1996). Abbott (1984, 1992b) suggests that to overcome in this issue for the practical purposes of research, the difference between occurrences and events should be taken as only relative. At the highest level, the case itself (i.e. the subject of the historical account) can be seen as a very complex event. It follows that

the researcher must first decide what the single-case historical narrative is a case of. Then the researcher must specify the context within which the case is meaningful: the unity of the case must be set against a contextual background that justifies treating it as one enduring case over a period of time (see Abbott, 1997). This not only involves putting the case in context and defining its boundaries but also sets limits on how properties of the case are to be interpreted. For instance, in a study of occupational professionalization, the researcher must define the success or failure of the effort on contextually-specific meanings (rather than standards taken from outside the narrative). Once the case is contextually delimited, the researcher can move to identifying the events.

Abbott notes that, in many cases, single-case narratives involve some sort of transformation (e.g. how a craft became a profession). The researcher must therefore identify the set of events that collectively would bring about the transformation. This has two implications: first, what made the transformation suitable for application of the theory should be traceable in the events. Secondly, there must be a successive (but not necessarily progressive) chain of events that move the plot to a final state. Put differently, theoretical matters of interest should be reflected in both the events of the narrative as well as their sequence. In this sense, historical narratives used in social sciences is always to some degree teleological (see Dray, 1989: 37-53). This is inevitable because it is possible to compare the chain of events with the causal explanation provided by the theory only if the events of the narrative can be linked to the analytical focus of the theory (and its assumptions about what has happened). What is important in this respect is that the explanation provided does not leave out any significant event or set of events that could challenge the explanation. In other words, the narrative and explanation fit when additional historical detail is 'saturated' by theory.

Following Abbott's advice, I have identified the different levels of analysis at which events of interest can occur and start the data gathering at lowest levels. For instance, formation of groups among workers can occur in a local labor market or among the employees of the same company at office level. This is a level at which the story can have unambiguously coherent subjects. But for this event to be considered consequential there must be a link (predicted by theory or provided by history) to the higher levels of analysis (the link may be constitutive or causal). For example, changes in employment relationship of one company can be indicative of changes in employment relationships in the industry or, if the evidence supports, a cause of those changes. Obviously, events at the lower levels can influence events at the higher levels and vice versa. It is important, however, to note that apparently similar events at different levels may have different 'stories' and therefore little direct relationship. What is important is to establish the relationship between events at different

levels of analysis (agents, locations, etc.) when the narrative moves from one level of analysis to another.

A more complex issue for colligation within and across levels of analysis concerns the temporal dimension of events. Any historical study of institutions must be sensitive to timing and sequence of events and conjunctures between them (Pierson and Skocpol, 2002). It is important to address the temporal dimensions of events directly because usually events at lower or higher levels are partially simultaneous and, perhaps, only analytically distinct. The decline of IBM in the 1980s is attributed to emergence of bureaucratic rigidities within IBM, change in IBM's selling strategy and competition from the cloning manufacturers. Of these three, the first two occur at the company level but one (bureaucratic fossilization) is a long-term event and the other is short-term. The third (external competition) is, compared to the other two, a medium term event at the industry level. The decline of IBM itself is an industry-level event, not a company-level event (a sole company cannot rise or decline). Whether it is a long-term or short-term event depends on the context of other simultaneous events which were occurring at the same level or higher. It's end, the 'comeback' and stabilization of IBM's position in the industry, can only be marked ambiguously but this only matters if the larger story depends on it. For example, in my research, which is not concerned with the history of IBM (or even the relative power of companies) per se the ending of this event is less important than the state of the competition that ensued (the redistribution of power in the industry) which becomes a structural link between the decline of IBM and other events in the industry (such as rise of Microsoft).

#### **4.3. Reflections on my research**

Having explained the methodological foundations of colligation and causal process tracing, I now reflect on my own research. I begin by elaborating my explanatory framework based on the advice of Abbott. Then I will try to clarify the assumptions of my research and the main predictions that can be derived from my theoretical framework. I will conclude by making a note about the presentation of material in the historical narrative.

My research is focused on understanding the historical processes that have led to differentiation of IT occupations. As such, it is a case of study of occupations within an institutional context. I presented a theoretical framework for understanding this phenomenon in chapter three and identified its context as that of the second-half of twentieth century work in America. In the table below, I have presented elements of the theoretical explanation in the way that it could be used to construct an explanatory narrative. For each element of theoretical explanation, I have specified the actors that are involved in shaping

and the main levels of analysis in which their interactions can be understood. Interactions among employers occur at the level of industry. Interactions between workers and employers can occur at sub-company (a division or a specific office/plant) or company level as well as company groups and the industry. Among these, sub-company and company groups are more transient levels in the narrative and are only significant if they become part of (or have a lasting influence on) one of the levels mentioned in the table.

Elements of the theoretical explanation		Actors involved	Event level of analysis
Market Structure	State of competition	Employers	Industry
	Vertical disintegration	Employers	Industry
	Decisions of dominant companies	Employers	Industry
Labor Market	Size (number of workers)	Workers, employers	Company, localities, industry
	Entry conditions (de facto)	Workers, employers	Company, industry
	Training providers	Employers, workers, other training providers	Company, locality, industry
	Worker Collectives	Workers	Company, locality industry
	Employment relationships (norms and practices) and decisions	Employers, workers	Company, industry
Sociopolitical Environment	Regulatory framework	State, employers, workers	Industry, economy-wide
	State actions	State, employers	Industry, economy-wide
Delivery Practices	Object and platform of IT work	Employers, workers	Company, industry
	Value contribution of IT work	Employers	Company, industry

Table 4-1: These elements will be used to structure the narrative.

The main argument of this study is that, in the absence of favorable government policies towards workers, the strategic and tactical decisions of corporations involved in the dynamics of market have destabilized IT jobs and effectively prevented the formation of a solid occupational collectivity that would lead to institutionalization of jobs. The building of this argument relies on analyzing the history of IT occupations according to the terms specified in the explanatory framework presented above. Following the principles of process tracing described above, I now reflect on the various aspects of the theoretical expectations that I can base on my theoretical framework.

A central tenet of my research is that, in the historical period that is of interest in this study, jobs cannot be institutionalized without the formation of collectives of workers. I consider this to be a necessary condition for the institutionalization of jobs in the contemporary era.

In the beginning of the twentieth century, however, many manufacturing jobs in the U.S. were institutionalized without prior formation of worker collectives because companies were interested in ‘rationalizing’ their fragmented employment systems (Jacoby, 1985). But the explanation for this period is not radically different from the one used for the contemporary period. Companies’ interest in rationalizing their employment system was attributable to the effects of wars and government intervention in regulating the labor market.<sup>9</sup> But in an environment in which government does not intervene in the labor market (or does only in favor of employers) and employers do not care for stable jobs or employment royalty, formation of collectives by workers is necessary for institutionalization of jobs. These collectives could be along occupational lines, employment relationships or the like. The important issue is whether or not they can stabilize one or another aspect of jobs. Therefore, in my study, I will focus on formation of collectives while following the causes of differentiation of jobs at various levels.

Note that in the argument specified above, the necessity of formation of collectives depends on absence of other factors. Therefore, the narrative must provide evidence that the other factors (like direct government intervention) were not present. It is also notable that within my theoretical framework I do not make any assumption about what encourages workers to form a collective. It could be shared grievances against an employer, competition from other groups (potential entrants) in the labor market, petition to government, etc. For my research, these developments are interesting if and only if they result in the formation of a collective and therefore become subsumed under the story of that collective. Therefore, I will be looking for the widest possible range of factors that could lead to formation of collectives and see if they had such an outcome. This could be at the level of specific offices, localities and higher (see Van Mannen and Barley, 1986). At the same time, for each collective of workers identified, I will be looking for the range of factors that had led to it and whether or not they had an effect. But the formation of collective is only the first step towards institutionalization of jobs. In the words of Dietrich Rueschemeyer (1986: 76):

“The simple fact that a collective organization exists is no guarantee that collective interests will be pursued. ... [T]he organization has to take the initiative in the crystallization of common interests and in the pursuit of strategies; the organization has to assume a leadership role in order to serve common objectives.”

If the organization succeeds in mobilizing and unifying its members, then it is likely that is influential in relations with other actors, including employer(s) and the government. When

---

<sup>9</sup> Many unions gained power *after* jobs were institutionalized and subsequently increased pressure on employers to preserve workers’ benefits and securities. Their power lasted only until early 1950s and the bureaucratic employment system itself gradually disappeared after 1960s, exactly the period in which IT jobs raise in number and significance.

aggregating the outcome of these interactions, the general prediction is that the alignment that is more powerful is more likely to be successful in shaping the outcomes in their own favor.

I have demonstrated that building explanations based on historical narrative is more like weaving than building: initially theory informs the outlook and focus of research and data selection. Incorporation of further detail about the study has led to elaboration of theory in greater detail, which in turn shapes the historical account to be presented and examined. In presenting the historical material that I have gathered, I have organized them according to the explanatory framework that was built on the basis of my theory. At the end of each episode, I provide a summary of main developments in that episode, which will be the basis of narrative in the next episode.

#### **4.4. Ontological and epistemological foundations of my research**

Do events exist ‘out there’ or are they ‘fabrications’ of historians? Are sequence of events a matter of chronology and time-keeping or is it the historian that orders events and gives them beginnings, middles and ends? Historians and philosophers of history have debated these issues for centuries. It was Auguste Comte’s ‘positivist conception of history’ that gave rise to a positivist orientation towards ‘the past’. Since then varieties of relativism has appeared (Kosso, 2009): some believe all there is to the past is determined by us (ontological position), some believe that all we can say justifiably about the past is determined by us and our (implicit or explicit) theoretical outlook (epistemological position), and still others believe that our capability to speak about the past is determined by our language (post-structuralist position). Without entering into details of this debate, below I present the assumptions that support the foundations of this study.

The most important point to be realized about ‘the past’ is that it is not empirically available and therefore it is not possible to hold on to an empiricist standard of correspondence when discussing history. So any account of the past is partial, no matter how it is constructed. But the fact that accounts are partial does not mean that they are fictional (cf. White, 1992). It is possible to construct narratives without necessarily nullifying its truth-value. This is because historians are rarely (if ever) alone in their constructive task: they share a universe of references with their audience, possibly including those who have lived through the past being narrated. More specifically, historians’ narrative must be comprehensive in the sense that it must include the minimum amount of historical material that other historians have found relevant to the subject. A historical account, therefore, is never the final word on the matter; they will be subjected to the various criteria of historians in ways that a work of pure

fiction can never be (see Norman 1991). In a more fundamental sense, as secondary sources, they become part of the past which is used by historians (Marwick, 2001). So, the past may not be accessible but history is not entirely subjective or relative.

Social scientists interested in history must similarly follow the historiographical standards of historians. Surely, as it was described above, the accounts they give are couched in specific terms and focus on specific events. Events and their relations are as much constructed by the historian as given by the past but as long as they allow the reader to make sense of the events independently of the explanation given (that is, they do not overdetermine their account by their theory), their narrative can be evaluated as an account of the past. In fact, a sign of robust process tracing, if not challenging a proposed theory, is uncovering of hitherto unknown causal processes which implies discovering formerly unimagined and untheorized chain of influence between events. It may also reveal *ceteris paribus* assumptions that lay behind the theory. This points at the possibility of a critical realist position<sup>10</sup> towards understanding society through the past: as research into the records of the past reveals more and more about the lowest levels of colligation and analysis in the narrative, the explanation can be assessed again and again. In the words of Andrew Sayer (2000: 144):

“...if an analysis of a specific case leads us to misspecify local history, then the synchronic structures [(intra-level relations)] and mechanisms must have been wrongly inferred, and, if an historical, explanatory narrative misconstrues the explanation of the local sequence of events, its implicit analysis of the configurational dimension must be wrong.”

From this perspective one can distinguish between the real (objects -like individuals, roles, corporations, institutions, etc.- and their causal power or capabilities and the structures within and between them), the actual (mechanisms that are generated by activation and interaction of capabilities in *real* objects) and the empirical (what can be observed about the actual, like the records of ‘the past’). Submitting to such a stratified ontology allows the researcher to make a sharp distinction between causal power and causation (or activation of that power) (Sayer, 2000, see also Archer, et al. 1998). It also recognizes the possibility of emergence of multi-layered and multi-aspected social phenomena. Thus, objects can be emergent from structures and/or a constitutive part of other structures and therefore, can be conceptualized and observed by the researcher in many different ways.

---

<sup>10</sup> It is important to note that George and Bennett agree in principle, if not in details, with Sayer’s position (see George and Bennett, 2005: 141 cf. Sayer, 1993).

#### **4.5. Constructing the objects of study**

Having specified the methodological and historiographical principles of my study, in this section I will construct in more detail the core object of my, i.e. the group of workers that I am going to study. I will first delineate the relevant ‘task area’ (alla Abbott, 2005) that defines this group of workers. Then I discuss the geographical limits of study, at both nation-level and the specific regions that will be the focus of study. Finally, I describe the temporal boundaries of the study and its periodization.

##### **4.5.1. *The object of study: software work***

I have mentioned the difficulties of drawing boundaries around IT workers in the previous chapter. I clearly could not rely on pre-determined job titles, educational backgrounds or professional memberships as proxy identifiers. Therefore, I have to delineate a group of IT workers that is analytically relevant, historically identifiable and, for the purposes of this research, manageable. I have already noted that my focus is on the IT industry and that I exclude all forms of blue-collar work in IT (including hardware manufacturing but also industrial production of shrink-wrapped packaged software). I also exclude from my analysis ‘white-coat’ hardware engineers, those whose primary job is to design and develop physical (microelectronic) components of IT and draw primarily on the knowledge-base of electronic and material engineering. In this section, I discuss the boundaries of the object of my study, namely “software workers”, which I define as the group of workers that engage in activities related to the design, development and use of software and services (i.e. the intangible aspects of IT) in the IT industry.<sup>11</sup>

In order to justify this definition, one must clarify the notion of software and software work. It is commonplace today to define software as the set of instructions that tell the computer what to do or, put more accurately, direct the actions of hardware (e.g. Campbell-Kelly, 2003: 2, Messerschmitt and Szypersky, 2005). However, such a convenient conception hides the several twists in the meaning of software that have occurred over the past decades. The term “software” was originally used (in written form) by John Tukey (1958) to refer to “interpretive routines, compilers, and other aspects of automative programming” as opposed to hardware (which comprised of “tubes, transistors, wires, tapes and the like”). This somewhat open definition was upheld by the wider computing community for whom software denoted everything related to computing that was not part of the hardware, including computer programs, programming tools, personnel and operational procedures

---

<sup>11</sup> As opposed to IS workers that are employed in user organizations (as in Friedman and Cornford, 1989). I also consider the IT workers in management consulting firms as IS workers, principally because they have not had a visible impact on the history of IT industry.



(see Ensmenger, 2010: 7). In the mid-1960s, software had different meanings in different contexts. In some contexts, it was closer to Tukey's notion and what today is known as systems software (see Mahoney, 2002: 43), while in others, like "software houses", it referred to all sorts of programs but no longer the personnel or the procedures. By the mid-1980s, when personal computer had entered workplaces and homes, most people encountered software as neatly packaged products that they bought and installed (see Campbell-Kelly, 2003: 2).

This brief and rather sketchy discussion of variations in the meaning of software is important from two aspects. First, it highlights the fact that historically software has meant different things to different actors or groups. A researcher interested in the historical developments of software work must be attentive to these variations in order to correctly identify software workers in different contexts. At the same time, it is important to note that "software" as a term has retained its utility: it is possible to find historical continuity between different 'things' that were considered as software in one or another era and therefore to demarcate a certain task area as "software work" in a way that is meaningful and illuminating.

This leads to the second aspect: one can analytically separate 'software as technology' from what was considered as software by different groups in different times. Software as technology can be understood as the product of efforts towards abstraction and automation of logical operations that otherwise would have to be done by hand and using wires (aka. rewiring, Cowan, 1997: 297-8; Haigh, 2002). The move from manually configuring hardware to automatically processing abstract 'syntactic tokens' to the same effect (Kallinikos, 2011: 4) has been the foundation of creation of software as the intangible and yet crucial component of computing devices. Yet, achieving this has only been possible by simultaneous formation of skills and identities that shaped subsequent developments and very often reproduced not only the distinction between software and hardware, but also skills and identities associated with each of them (Jesiek, 2006; Nardie, 2011).

Thus when I exclude hardware engineers from my study, I am fully aware that early programmers and software workers (like operators) had not only a working knowledge of the hardware but also were actively involved in its maintenance. However, the boundary between an engineer as designer of hardware and a similarly educated person as part of software team was meaningful and real. At the same time, I do not hold a constant conception of software as the basis of software work. Instead, I follow the narratives of existing histories of software industry and software work (e.g. Campbell-Kelly, 2003;

Ensmenger, 2010) and pay particular attention to what constitutes software and software work according to the actors in those narratives.

Of course, the actors involved did not always agree about what constituted software and at some points no one had a clear understanding of how to define software (IBM's unbundling decision comes to mind). This is why understanding software as a set of instructions is inadequate for this study: it hides the ambiguities and contingencies of software as a technological artifact, as a product of IT industry, and as a part of the task area of software workers. It is notable that, according to Ensmenger's estimate (2012), the actual writing of code was "no more than a third" of the time and effort put into development and deployment of any typical software. By adopting a broader and more flexible conception of software and software work, this study can provide a picture of varieties of software work that is historically more accurate.

This approach also brings into the view one of the less-noticed aspects of software that make it unique. Software is often considered unique for the technological or economic features it has (e.g. it is intangible and virtually free to reproduce) or for its interactions with other technological systems (for instance, advances in hardware have dramatically increased the processing power while reducing the costs of executing software). What is often unrecognized is that software (in the broadest sense) is the only major (indeed ubiquitous) technological artifact that, while playing a vital role in economy and everyday life, is designed and developed by an undifferentiated and unrecognized group of people (compare with buildings, drugs, cars, etc.). While many in this task area refer to themselves as scientists or engineers and are generally considered as "educated" (e.g. Evans, et al 2006), the truth is that many of them are (or at least entered this area of work) as amateurs learning on the job. This not only has made the thought of regulating this area of work unthinkable, but also has rendered any comprehensive understanding of the institutional arrangements of production of software impossible. (On the "histories of computing(s)", see also Mahoney, 2005). By slicing the domain of work that is of interest, this research can aim at providing a comprehensive picture of a small but important part of IT work.

This approach results in systematic exclusion of software components that are embedded in non-IT products and sold outside the IT product market (automobiles, missiles, washing machines, etc.). Moreover, I exclude the work performed on software components that are embedded within IT products once those activities came to be considered as part of hardware work rather than software work (e.g. microprogramming).

This is consistent with the occupations that are often studied in the literature (e.g. Levina and Xin, 2007, Ensmenger, 2010: 12). Nevertheless, this conception of software work is, at the same time, broader and more specific than those used in the literature. It is broader, in the sense that by choosing not to specify the list of occupations that are of interest, I keep the range of related occupations open. On the other hand, it is more specific because it only considers software work as a part of IT industry (rather than in all industries).

The reasons for focusing on a specific group of workers should be clear. The pragmatic reason is that it is necessary to narrow down the study to make the research project realistically feasible within the limits of a PhD. The theoretical and more important reason is that this group of workers has been more directly involved in the dynamics that I have described in chapter four. By narrowing down to IT producing companies, I can clear the view by bringing some order to the institutional context of software workers and, simultaneously, use my theoretical framework to explain developments in software work.

It is also clear from the discussion above, that it is not possible to narrow down the study to companies that exclusively produce software and software-based services (e.g. ‘software publishers’ and contractors) for a couple of reasons. First, companies do not maintain rigid identities and switch between activities quite regularly in the IT industry (e.g. Novell was a manufacturing company before switching to software, whereas Google started as a software-services company but now produces hand-held devices too). Secondly, a significant part of software markets and developments therein have been a result of decisions and activities of multi-sector IT companies (like IBM). Leaving these companies out would lead to missing some of the most important changes in the industry in terms of dynamics of market structure as well as IT skills.

It must be noted that I am not assuming that a real divide exists between software workers (in the sense used above) and similar workers in other industries. A programmer can move from a bank to work for Google and vice versa. But by quitting a job in the financial industry and accepting a job in the IT industry, he becomes involved in a different institutional context, at least as far labor markets and employment relationships are concerned. It is important to note that, as I will show in my historical narrative, in the early years of IT industry, software workers were placed in-between IT-producer and user companies.<sup>12</sup> The more important question is whether software workers (as defined here) could form a real social group. I believe they could, because at least in the early years, they

---

<sup>12</sup> Hebden (1975) discusses the “popular conception of DP as a single occupation” and its academic adherents (including, among others, Enid Mumford).

were a relatively small group of workers that had a highly valuable skill. In any case, the multi-level design of this research allows for identifying occupational groups within this group of workers, regardless of whether they were connected to a larger social group or not. At the same time, the theoretical model that I have used allows me to understand and explain the divisions that emerge in this group over time.

#### ***4.5.2. The geographical boundaries***

I have already mentioned that this study is focused on the U.S. and this has been incorporated into the theoretical framework and research design. The reasons for choosing U.S. are multiple: first and foremost, software work, as conceptualized here, had its roots in U.S. Other countries (with the possible exception of U.K.) lagged behind. Second, not only U.S. has had a pioneering role in the genesis of IT industry generally, it has had a dominant position in software innovation as well as software market. The U.S. software industry is the largest in the world (in terms of both sales and employment). The dynamics of software work in U.S. cannot be ignored even if one decides to study it in another context. Third, compared to the rest of the world, the U.S. IT industry has attracted much more attention from historians and scholars from different disciplines and therefore, historical data and reports about it are much more abundant.

Nevertheless, excessive insistence on remaining within geographical boundaries in an age of global competition is likely to provide a skewed picture of developments. On the one hand, American IT industry entered the market of other countries quite early in its history of development. On the other hand, it has faced competition from global competitors at least since the 1980s (the rise of IBM-PC cloners). For the purpose of this study, I assume that in terms of dynamics of expertise or formation of skills we can generally ignore what happens outside the borders of the U.S. Very few influential software innovations occurred outside U.S. (e.g. the ERP system of SAP AG, see Campbell-Kelly 2003: 165-167). I also assume that activities of foreign software companies inside U.S. can be to a large extent ignored because they had very small influence on the dynamics of market, expertise or employment.

Given the geographical distribution of software work, it is quite obvious that it is impossible to track down every single software worker (or even software firm). Unlike hardware manufacturing plants or purpose-built labs, software work can take place in almost any office building and hence, it is much more difficult to pinpoint where software activities take place. To the extent that this study is focused on layoffs and employment practices of companies, the geographical distribution of software activities create some limitations. This

is partly because most companies do not provide detailed information about their recruitments and layoffs (and workers and jobs involved).

This problem can be overcome to some extent by focusing on local news outlets and other sources of information that provide information about changes at the local level (e.g. regional labor markets<sup>13</sup>). This on the other hand, requires knowing where to look for information, at least at the level of state. I have tried to find these localities using three sources: 1) historical data about the gradual growth of firms (e.g. IBM opening offices on the west coast) 2) national statistics about where software-related IT workers are employed 3) few studies about “application districts” (e.g. Egan, 2000). Since the primary focus of the study is on dominant firms, I have paid special attention to the geographical states in which they were founded or in which they concentrated most of their software activities (e.g. New York/Massachusetts for IBM, Washington for Microsoft, etc.). I have tried to juxtapose information about these areas with national level information and specific locations in a few other states throughout the study to show the influence of reported dynamics beyond the above locations.

#### ***4.5.3. The beginning, the end and turning points***

The history of software work as conceptualized here is coterminous with the history of computer. One could argue that using the term ‘software’ for the period before the 1960s is ahistorical because the term was not used at the time (Jeisek, 2006). Nevertheless, I will use the terms software and software worker for the whole period of time between 1945 and 2001. This will be in line with the conventions that have emerged in the history of IT industry.

The history of IT industry starts with the increasing use of computers in businesses in the post-Second World War period and follows through the 50s as a small ‘programming services’ sector in the computer industry. The major turning point, by all accounts, comes in 1969 when IBM decided to charge its customers for hardware and software and services separately. Even though this even has been called ‘the crucial inflection point’ in the development of software industry (Yates, 1995: 123) A significant rise in the sales of ‘independent vendors’ (i.e. specialized software companies) does not arrive until more than a decade later, with the arrival of PC (in 1981). But rather than taking arrival of PC as the

---

<sup>13</sup> Wherever regional is used in this study it refers to a geographical unit composed of multiple districts within a state or across several states. The exact boundaries will be specified in the context. Local, on the other hand, is reserved for district or city level boundaries.

turning point, I focus on the turning point in the power of IBM in the industry: in 1985, IBM's growth stalled and then IBM began to decline.

At the same time that IBM fell from its dominant position, Microsoft became the new dominant company. Even though it never near the level the dominance that IBM had (it took 14 years from 1985 for Microsoft to supersede IBM's software sale), it became the most important player in the IT industry. The second half of 1990s is marked by the Y2K problem, increased use of internet, and the dot-com bubble all of which are associated with significant changes in software work and software labor market. The end point of study is 2001, just after the burst of dot-com bubble. I will use the insights from the study to analyze the developments since 2001 in the discussion that follows the historical narrative. Therefore the periods covered in this study are as follows:

1945-1954: The rise of software work

1954-1969: The rise of IBM

1969-1985: An IBM world

1985-2001: Microsoft's rise to power

Now that I have specified the object of study, its boundaries and its periodization, I can discuss the sources that I have used.

#### **4.6. Notes on the sources**

Several accounts of the history IT and IT industry inform my main narrative. Aspray et al. (2013) and Ceruzzi (2003) provide informative accounts of the parallel developments in the history of IT and the IT industry. Yost (2005) and Campbell-Kelly (2003) provide more detailed accounts of, respectively, the computer industry and the software industry. Apart from these historical accounts, I have relied on two books for further detail and analysis: Fisher et al. (1983) and Flamm (1988). Since IBM had called on Fisher and his associates in its defense in the antitrust suit, whenever discussing the issues of competition I have tried to use alternative sources for corroboration. Government and industry reports were similarly helpful in acquiring a broad perspective on the developments in the industry. Collectively, these sources were used to construct the accounts of object and platform of IT work, market structure, training providers and the size of labor market.

Material about individual corporations has similarly been gathered from corporate histories as well as biographies and memoirs of businessmen and IT workers. I have also relied on oral histories produced by the Charles Babbage Institute for histories of specific companies. Information about employment relationships of organizations has been gathered from these sources and, whenever possible, studies of employment relationships in different companies.

Ensmenger (2010) has been the basis for my narrative of the early decades of software work. Data about worker collectives that were not covered in the sources mentioned above were gathered from other sources. These include newspapers, trade journals, company archives and similar sources. For access to these sources, I have relied to a large extent on the vast amount of sources that is available on the Internet. Online digital archives like archive.org have preserved extended company archives and reports. I also relied on digital copies of majority of newspapers (with the notable exception of *Datamation*).

I used national-level newspapers as sources for information about large companies and the industry. The main four newspapers that I relied on were *New York Times* (especially for IBM), *Los Angeles Times* (for Silicon Valley companies), and *Chicago Tribune*. I also referred to *Philadelphia Enquirer* and *Seattle Times* and *Wall Street Journal* for specific periods or stories. I also occasionally used weekly magazines such as *BusinessWeek* and *New Yorker* and *Ebony* magazine. For gathering more focused data I drew on several local newspapers, particularly in the Hudson valley. Fulton History online archive of newspapers provided access to the majority of these newspapers (such as *Binghamton Press*, *Schenectady Gazette*, *Pine Plains Register Herald*, etc.). *Kitsap Sun* and *San Jose Mercury* were sources of local news for, respectively, Seattle and Silicon Valley.

Beyond these general news sources, I relied on three principal trade journals: *Computerworld*, *InfoWorld* and *Network World*. *PC Magazine* and *Datamation* provided additional sources. I also relied on *Computer Resellers News* for the later developments in the PC industry. Beyond providing news about companies (layoffs, restructuring, acquisitions, etc.), these journals constituted the main resource for following the debates between IT workers, analysts and commentators.

In all these sources, I explored the job ads over a period of three decades (from mid 1950s to mid 1980s). The search was keyword-based and its aim was to uncover as many ads as possible. Several different words were tried for the same period. The ads (some of which are reproduced here) provided interesting insights about jobs and their benefits and status. I also looked at many wedding announcements, engagements, obituaries of IBM employees (especially in the Hudson Valley) in search of clues about individuals' educational background and jobs. Although not formalized, these data sources provided interesting insights about the life of IBM workers in that period.

**Part Two**  
**Software Workers in the American IT industry 1945-2008**



## **5 The Rise of Software Work: 1945-1954**

### **5.1. Introduction**

While the technical origins of computer (the machine) and programming (the activity) are quite well-known, the historical origins of software work as an occupational field has remained (at least until recently) obscure. What Alan Turing and John von Neumann - among others- laid foundations of quickly became the specialty of a certain group of individuals (spread across many locations) who were paid to make computing machines work. They emerged from humble positions as operators of what seemed to be sophisticated calculating devices (for scientific purposes in universities or accounting purposes in businesses) to form a new breed of ‘organizational man’ who would soon become the most-wanted and highest-paid of all employees bar senior managers. The transformation of their domain of activity and its accorded status and rewards was so quick that few traces of it were recorded and remarked upon. Nevertheless, it is essential for our purposes to explore these origins in order to understand the significance of subsequent developments based on the limited data that is available.

Therefore I begin by discussing the roots of software work in the early computation labs in which computers were first developed near the end of Second World War. Then I will trace the changes that occurred after the war, focusing on the shift in the context of programming jobs and the tools that were available. Finally I will consider the role of the SAGE project in the training of early programmers as well as establishing the de facto criteria for entry to labor market. At the end of each episode, I offer my interpretation of the developments that occurred as well as their link to the subsequent section. At the end of the chapter, I provide a summary of discussions.

### **5.2. The Beginning: proto-software work in two early cases**

Before there were electronic computers there were human computers. Human computers were mathematicians who used their skills, along with mathematical tables, mechanical calculators and other tabulating devices to calculate solutions to formulated problems, usually in scientific or military contexts. They often worked under the direction of scientists and engineers who provided them with problems and suggestions about the method of solving them (Grier, 2005). Early electronic computers were devised to do what human computers used to do. But the machines did not directly or simply replace humans. For one thing, their functional utility soon transcended the realm of calculation to include ‘data processing’ and ‘informatics’. For another, in order to function properly, early computing machines required attendance of human operators. Once given the algorithm, they could

solve the problem, but they still needed humans to feed in the step-wise solution to be followed. Advanced mathematical skills of human computers, as well as their familiarity with calculation methods and devices and their ‘sense’ of final result to ‘check’ the output of computer against possible errors (based on years of experience) made them ideal candidates for the new job. It is not surprising therefore that human computer continued to co-exist with and occasionally accompanied and contributed to the development of computing machines (Ceruzzi, 1991; Grier, 2005). The gradual process of change, therefore, involved a continuous shift in the boundaries of what computers did and what humans did rather than direct replacement or ‘automation’, even though after some time human computers became virtually non-existent and forgotten.

The six ‘ENIAC girls’ who are now often regarded as the first group of programmers were all former human computers involved in the Ballistic Research Laboratory of the U.S. Army. As human computers, their job was to calculate and compile hundreds of mathematical tables for accurate shelling and bombardment of long-distance targets during the Second World War. More experienced human computers developed ‘plans of computation’ for other computers and supervised their work. The aim of top-secret “Project-X” (ENIAC development project, initiated in 1943) was principally to automate the hand computation (Grier, 1996). Still, turning the mathematical formula into machine-understandable format required human computers. Therefore, six top-performing human computers were selected and recruited as ‘coders’<sup>1</sup>: they were responsible for ‘translating’ the mathematical formula developed by higher-ranking scientists and engineers into ‘machine language’.

It was soon clear that planners could not foresee the problems that the computer would run into. Coders frequently found themselves making modifications to the numerical algorithm, often with support from the supervising engineers but increasingly independently. They also came to know and understand the working mechanisms and physical characteristics of the computing machine as well as, if not better than, engineers and planners. Therefore, as they became more and more involved in the process of applying computers to problems, they tried to distantiate themselves from the mechanical connotations of ‘coder’ and became identified as ‘programmers’. Some of their early innovations were not related to the implementation of any scientific algorithm but to the operational control of the computer itself (like devising “a stop instruction”, see Ensmenger, 2010: 37).

---

<sup>1</sup> They were trained by Adele Goldstine who was married to Herman Goldstine, one of the designers of ENIAC along with von Neumann, Mauchly and Eckert. She had a Masters’ degree in Mathematics.

Despite all this, Goldstine and von Neumann who were involved in the ENIAC project, laid out the principles of division of labor between ‘planners’ and ‘coders’ along supposed technical competencies. In a series of volumes titled *Planning and Coding for an Electronic Computer Instrument*, published in the late 1940s, they specified ‘planners’ as those responsible for formulating the problem mathematically, selecting a numerical algorithm and performing a number of analyses (like estimating precision requirements) to ensure that the interim results of algorithm will remain within the limits of calculative capacity of the computer and the final output of algorithm would be acceptable for practical purposes. The coders, on the other hand, were assigned to the last (and the least intellectually challenging) step: putting the selected algorithm into a machine-understandable form, either by punching cards or using switchboards and plugs. The fact that such a division of labor was untenable was not recognized yet.

Conditions surrounding the development of another early computer, the Harvard Mark I, provide a more complex picture. There, the “Automatic Sequence-Controlled Calculator”, as it was officially known at the time, was constructed and delivered (donated) to Harvard University by the IBM corporation in the summer of 1944. Similar to ENIAC, the purpose of this machine was to automate calculations<sup>2</sup> (Beyer, 2009: 36). Although installed in a university environment, Mark I was run by the U.S. Navy and most of the ‘crew’ were Navy personnel. It was designed by Harvard physics graduate and Navy Commander Howard Aiken who along with three assistants were now responsible for using it to solve the problems referred by other scientists and researchers. Two in their group, Grace Hopper and Richard Bloch<sup>3</sup>, were juniors in rank and were assigned to the task of ‘coding’ the computer while Aiken and his deputy Richard Campbell provided supervision and attended to the problem of devising new machines. Despite the terms used, Bloch and Hopper’s job encompassed all of the steps mentioned above: they had to work with scientists from different disciplines to arrive at a mathematical formulation of the problem, select an algorithm to solve the problem and translate them into a machine-readable form. They also needed to know “the hardware of Mark I in all its intricate detail” (Beyer, 2009: 45).

Over time, a more elaborate organizational structure emerged. A technical feature of Mark I triggered a different division of labor; one which is more familiar (at least in technical terms) to our understanding of software work. Coding the ENIAC meant setting up switchboards and plugging or unplugging wires by hand. For each run of programs, coders

---

<sup>2</sup> ENIAC stands for ‘Electronic Numerical Integrator and Computer’. AC in many early acronyms stands for automatic computer; a clear sign that these machines were aimed at simply automating hand computations. Other examples include: UNIVAC, JOHNNIAC and ILLIAC.

<sup>3</sup> Both had degrees in mathematics (Hopper, a Ph.D. and Bloch a bachelor’s degree).

had to modify the switchboards and plugging. In contrast, Mark I relied on an electro-mechanical mechanism for inputting instructions that would ‘read’ continuous punched paper-tapes until the job was done. Therefore, there was no need for human intervention between runs. The coders were responsible for producing reliable instructions which could be translated in the form of punched paper-tapes. Thus, the task of punching the paper-tapes, feeding them into the machine and operating the machine was left to ‘the operators’ (Beyer, 2009: 56).

Operators were also responsible for making certain manual modifications in the machine from time to time and were also involved in the ‘testing’ of newly devised instructions. In order to do so, they relied on ‘operating instructions’ which provided step-by-step guidance and some troubleshooting tips. The re-usability of punched paper-tapes also provided incentives for thinking and coding increasingly in terms of ‘subroutines’. Embedded instructions and even hand-written commentary on paper-tapes to guide the operators also became common practice (Beyer, 2009: 63). These practices could have resulted in blurring the boundary between coders (later, called programmers) and operators. But in reality, this never happened. As a clear sign of distinction, during the fifteen years of operation of Mark I, the coders were always Navy officers while operators were enlisted lower-rank personnel.

### **5.3. Analysis**

I have focused on these two early sites of computer work because of both their precedence and their influence on later developments. As such they shall serve as two cautionary tales from the earliest years of computing that warn about the complexities of the history that remains ahead. Technology does indeed play a role in the organization of software work but that role is always limited. Moreover, it could provide grounds for unification of ranks and responsibilities as well as their diversification and distinction. Indeed, as the students of relationship between technology and organization of work have repeatedly shown, technological change provides ‘occasions for structuring’ roles and responsibilities (Barley, 1986). Whether or not change happens and the shape it takes is contingent over several other factors. We saw in the case of ENIAC that even though conflation of planning tasks and coding tasks was common practice and contributed to a rise in coders’ status and responsibilities, the boundaries were inscribed in guidebooks rather than removed. Thus, from ‘blue-cross science’ (Grier, 2005: 5) to ‘glorified clerical workers’ (Ensmenger, 2010: 35), programmers were viewed as a low-rank group of white-collar employees.<sup>4</sup>

---

<sup>4</sup> Haigh (2001) posits early systems analysts in a similar situation (see below).

There might have been several reasons for this. Historian Nathan Ensmenger points out that in ‘all’ pioneering computing projects, “programming was considered, at best, an afterthought” (2010: 29). The presumed simplicity of the task (in an environment full of top-class scientists and engineers) relegated programming to the lower levels of organization. Ensmenger also suggests that predominantly male scientists and engineers considered coding a low-status work in part because it was associated with the gender identity of human computers and other low-rank clerical workers such as key-punch operators:

“In any case, the historical pattern of the nineteenth and twentieth century has been that low-status occupations, with the exception of those requiring certain forms of physical strength, have often become feminized. [The same notion was reinforced in case of programming which was viewed as] more handcraft than science, more feminine than masculine, more mechanical than intellectual.” (p. 38, see also Light, 1999)

In few cases like Mark I, where women enjoyed a relative equality, other organizing principles came to play.<sup>5</sup> In the specific case of Mark I, the laboratory “was embedded in a differentiated organizational structure that borrowed from shipboard and factory operations” (Beyer, 2009: 61). Moreover, Aiken managed the laboratory like a commander rather than a research director and required everyone (sometimes fiercely) to adhere to the chain of command and disciplinary principles. Thus subsequent technological developments (including Mark IV which was completely electronic) did not alter the organization or division of labor in any significant way.

These two examples also illustrate how similar job titles can have very different meanings in different contexts. Here and elsewhere in this study, job titles act more like ‘moving signposts’ rather than ‘fixed markers’. They can help trace divergent developments of particular subdomains of software work without necessarily forming a consistent framework. As we know, the distinction between coders, programmers and operators persisted over the next several decades even though none of these terms ever found fixed or commonly agreed meanings. While some may argue these three job-categories (along with systems analysts<sup>6</sup>, see below) reflect the technical necessities of division (and organization) of software work in its early stages (e.g. the traditional Systems Development Life Cycle), recorded evidence suggests:

- 1- from the start, the ‘technical’ component (environment) of programming did not provide a unified scheme for usage of terms like operator or coder; and

---

<sup>5</sup> See Abbate (2012) for a fuller exploration of role of gender in the history of programming.

<sup>6</sup> An equally ambiguous term with roots in wartime mathematical programming and accounting departments. Increasing use of computers for commercial purposes, extended its domain to almost any task involving understanding the application domain of computers.

2- the organization of work and allocation of responsibilities (including demarcation of jobs) were shaped -at least equally- by the social environment.

Still, what is of interest to us is neither the naming conventions (or fashions) nor the local conventions and situated settlements. We are interested in understanding how the way different instances of IT jobs were defined, valued and dissolved changed historically. We know, for instance, that in both of the projects mentioned above programmers or coders were selected from among the human computers (or more generally, women<sup>7</sup> with some background in mathematics). The explanation for why and how this happened must be sought in the historical circumstances.

As I mentioned above, part of the reason why women were admitted into programming (or more specifically, coding) jobs was that scientists in charge saw it as a low-status, mechanical and largely unproblematic task like other clerical jobs and ‘appropriate’ for women. The supply of mathematically-savvy women, on the other hand, was itself a product of changing norms about the status of women in society and particularly their education. Since the early decades of the twentieth century, women had entered university programs in mathematics largely because it was the only socially acceptable science degree for women (Physics degrees were also available but to a lesser extent). After graduation, many of the graduates who sought a job became maths teachers or human computers (see Grier, 2005: 3-4; Golemba, 1995). When U.S. became engaged in the WWII, the number of men in office jobs in army decreased while the demand for human computers multiplied. Subsequently many women were employed as human computers.

Among the ENIAC girls two did not have a degree in mathematics. Betty Snyder started university as an electrical engineering student but soon changed subject and graduated with a degree in journalism. Marlyn Wescoff had a major in Social studies and English and a minor in Business. Wescoff states that she was hired because she “knew how to run a calculator” (Fritz, 1996: 22). However, Lila Todd (another ENIAC girl) points out that “when women math majors were no longer available, women college graduates with other majors and some mathematics were hired” (Fritz, 1996: 15). The demand for human calculators had pressed the Ballistic Research Laboratory supervisors to reduce their qualification requirements.

Nevertheless, this was a short-lived phenomenon. By the end of war, not only the demand for human computers was sharply reduced but also many female job-holders were replaced

---

<sup>7</sup> For instance, at the time ENIAC was being constructed, there were 80 women and three men in Ballistic Research Laboratory (Fritz, 1996: 15)..

by returning veterans (Light, 1999). Still, women were recruited as coder/programmers in many computer labs in the immediate post-war years<sup>8</sup> (Frtiz, 1996). Thus we can see that even though the organization of work in each project was different, a largely coincidental combination of particular social norms and gender-biased relations with the effects of war produced a relatively coherent pattern for the way in which proto-software jobs were defined, organized and recruited for. This initial arrangement, however, was soon disrupted by new waves of social change as well as emerging understanding about the difficulties of programming. These factors, along with the explosive diffusion of computers in businesses (see below), made programming far more important than a low-status clerical job that could be left to women. Gradually programming was defeminized, a trend which lasted for several decades.<sup>9</sup>

#### **5.4. Into 1950s: Farewell Mathematics, Hello Automated Programming**

At the start of 1950s, there were no common standards or practices for installing computer machines. Nevertheless, programmers were still a relatively small community who had regular gatherings and collaborated with each other over their problems. Following the concluding paper of the first Joint AIEE-IRE<sup>10</sup> Computer Conference in December 1951, a notice “was submitted in writing” which reflected the discussions about the “problems arising in programming” in informal sessions and conversations. Top on the list was the differences in “operating procedures” of various computers and the role of programmers and operators. After a brief listing of the positions of the various participants in the debate (each representing a computer lab), it was reported that “no conclusion was reached but it became apparent that this topic may well be a heated subject of controversy in future meetings” (see Forester, 1951: 113-14). It seems that at this stage, as Ensmenger remarks, programming was “intrinsically local and idiosyncratic”:

“each individual computer installation had its own unique software, practices, tools, and standards. There were no programming languages, no operating systems, no best-practice guidelines, and no textbooks.”(Ensmenger, 2010a: 58, see also p. 74)

Nevertheless, most of the programmers (men and women) had college-level mathematical degrees and were trained in programming at the pioneering computer projects of the late 1940s.<sup>11</sup> But the association of programming jobs with mathematical skills was also waning. Within the span of twenty years (early 1950s to late 1960s) mathematical skills

---

<sup>8</sup> Even though ENIAC was not operationalized until after the war (1946), women were recruited as early as 1943. Nevertheless, 4 of the original 6 had voluntarily resigned by 1951.

<sup>9</sup> Persistent sexist views and degradation of women’s jobs in IT continues to create significant challenges for professionalization of software work (Ensmenger, 2010b).

<sup>10</sup> American Institute of Electrical Engineers and Institute of Radio Engineers, respectively.

<sup>11</sup> A group of pioneering programmers participating at the UNIVAC conference in 1990 agreed that almost until 1950 none of them was hired as ‘programmer’.

moved from ‘absolutely essential’ to ‘preferred’ to ‘irrelevant’. Historians have ascribed the decreasing importance of mathematical skills in software work (and especially, programming) to two developments in this period: 1) ‘automated’ programming languages automated a lot of mathematical coding work that was previously done by programmers/coders. 2) computers were increasingly applied in non-scientific domains and therefore, focus shifted from scientific calculation to business data processing (Ensmenger, 2010a: 20 and Ch. 4 and Beyer, 2009: Chapter 8). In this section, I consider each of these developments and their implications for changes in software work in turn.<sup>12</sup>

Employment Opportunities  
in **IBM** for  
TRADE MARK  
**APPLIED MATHEMATICIANS**  
**PHYSICISTS**  
**THEORETICAL ENGINEERS**  
**PROGRAMMERS**

Outstanding Opportunities  
Permanent Positions

In solution of scientific and technical problems on electronic computing equipment for rapidly expanding division of IBM. Unusual opportunities for professional development and advancement.

Write giving full details including experience and education to:  
**Dr. C. C. Hurd**  
**Director of Applied Science Department**  
**International Business Machines**  
**590 Madison Avenue, New York 22, N. Y.**

Figure 5-1: IBM programmer ad, (Notice “permanent positions”) 1952

<sup>12</sup> It must be noted that despite all these transformations, mathematics became an institutionally-recognized part of software work (through occupational classification and computer science). For a discussion of the multifaceted relationship between software and mathematics see Mahony (2002).



The advance of automated programming languages is often credited to the efforts of Grace Hopper of the Harvard Computation Laboratory who had moved to Remington Rand (Campbell-Kelly, 2003; Ceruzzi, 2003)<sup>13</sup>. In her view, just as early computer machines helped automate hand calculations, the most repetitive part of programmers' job could be automated by using "a catalogue of subroutines" and "coded coding" using a 'compiler' (Beyer, 2009: chapter 8, see also Norberg, 2005: 195-7). It is notable that even though this idea underlies nearly all programming languages today, its very possibility was hotly debated in the early 1950s. A good deal of 'salesmanship' was necessary to sell this idea within the then-small community of programmers. Some argued that programming was essentially a human skill that was far too complex to be handed over to computers (see Hopper, 1981; Metropolis, 1989). Many felt threatened by the development of automatic programming languages because it made their hard-earned skills obsolete (see Beyer, 2009: 235-242).

Among managers, many (including those of Remington Rand) believed that the only function that computers could perform was calculating. To them, the idea that computer could program itself was absurd and any effort in this direction seemed like expensive experiments. Grace Hopper and her team had to use their spare time to build these languages. To convince others, they had two challenges to overcome. First, they had to show that computers could be 'educated' to program themselves and the outcomes were technically reliable (see Hopper 1952/1988). Second, an economic case had to be made to convince managers that writing a program that simplifies writing of other programs is a worthy investment. It took nearly three years and successful demonstration of three versions of the programming language (A-0, A-1 and A-2) to convince Remington Rand managers to dedicate resources (in the form of a new department) to the development of programming languages. By that time, many others in the industry and technical community had initiated efforts to develop programming languages. These and subsequent efforts led to the development of well-known programming languages such as Fortran (in 1957) and COBOL (in 1961). Soon, they were simply seen as the new tools of the trade and the question was whose (which vendor's) programming language was better (see below for a discussion of the competition).

There seems to be no record of the actual effects of advance of automatic programming languages on programmers' work at its early stages (see, for instance, Backus, 1980). There

---

<sup>13</sup> For another independent early effort, see Campbell-Kelly (1980). For different meanings of compilers in the 1950s see Backus (1980). For a general review of the concept see Knuth and Pardo (1977).

is little doubt that automatic programming languages removed the most repetitive and mechanical part of programmers' job, namely 'low-level' coding. In doing so, it improved programmers' productivity and ultimately eradicated the occupational category of coders. But the advance of automatic programming languages alone could not make mathematical skills irrelevant. At best, automatic programming languages would have availed programmers of a particular set of mathematical skills (see Ensmenger, 2010a: 65). In fact, in her original paper on the concept, Hopper claimed that UNIVAC (the computer for which A-0 was devised) "has a well-grounded mathematical education fully equivalent to that of a college sophomore, and it does not forget and does not make mistakes" (1952/1988: 281). But she also hoped that availability of automatic programming languages will let the programmer "return to being a mathematician". By this, she meant the more intellectual aspects of scientific programming that involved formulating the problem and finding a step-wise solution for it (p. 273-4)<sup>14</sup>. So, in terms of programmers' skill, automated programming mainly changed the level of mathematical skills that were needed.

On the other hand, programming languages did contribute to the expansion of computer use in the business world. One could argue that without the automatic programming languages, programming would have remained a hugely labor-intensive effort, far beyond the resources of even large corporations. This was not only a matter of cost, but also of speed. As Hopper wrote in her internal report for Remington Rand managers in 1953, because business applications changed frequently, without automated programming, effecting those changes in computers applications were impossible or uneconomical:

"A change in government regulations, city ordinances, union contracts, or requirements of company management, even a simple change, such as income tax exemptions, may invalidate not only parts of runs [(execution of program)] but whole runs in such a base problem" (quoted in Beyer, 2009: 244).

She also mentioned that competitors were initiating their efforts in this regard and if Remington Rand managers wanted to have any success in commercializing their products, they had to do the same. In the same report, Hopper suggested that automated programming will reduce the number (and associated costs) of programming staff. Given the acute shortage of programmers with adequate skills (see below), the potential savings were too much to be ignored.

So, it seems that the emerging irrelevance of mathematical skills had more to do with the changing context of computer use rather than advance of programming languages even

---

<sup>14</sup> In reality, she became involved in the development of programming technics to be used in the business environment (for her role in the development of COBOL, see Beyer, 2009: ch. 11).

though the latter contributed to the possibility of the former. I will consider the evidence about the actual benefits of using automated programming languages in businesses in the analysis section below. What is important at this point is that automated programming languages lowered the perceived costs of investing in computer systems and facilitated their commercialization.

To put this in perspective, by one account, there were only two digital computers installed in the United States in 1950 and both were used for government purposes. That number rose to 240 in 1955 and 5400 in 1960 (Flamm, 1988: 135). Most of these installations were in non-government organizations and for business purposes (Pugh and Aspray, 1996; Ceruzzi, 2003). As I mentioned in the previous chapter, in this period software was bundled with hardware sale and was provided to the client as part of free after-sale services. Several factors can account for the establishment of this arrangement (see Fisher et al, 1984: 23-25; Delamarter, 1986: 40)

- 1) From a technical perspective, writing code that enables the computer to perform desired calculations still seemed much simpler than designing and building the hardware which was a state-of-art outcome of engineering.
- 2) From the buyers perspective, by purchasing computer machines, they were already taking a major risk and they expected to receive a complete working solution rather than an expensive calculator that needed extra investment to work.
- 3) It was also in the interest of vendors to demonstrate to potential buyers that their clients were satisfied with the purchase.

In the accepted arrangement, after hardware was physically delivered and installed at client's office, a group of programmers were sent to client to train their employees and, jointly with client employees, develop software that was specifically tailored to client's procedures and needs. Clients, on the other hand, did not expect to hire significant numbers of new employees. After all, computers were expected to automate organizational processes and reduce clerical staff. The general expectation was that programmers (and their trainees) will program the computer without any major problem.

Actual installations, however, were far more problematic. For instance, a much publicized case is the first business application of computers in the United States, GE Appliances Division's UNIVAC in Louisville (KY) which was scheduled for delivery in January 1954. After physical installation was finally completed in August 1955, it took more than a year to produce a satisfactory payroll application and in the process, a whole team of programmers

were fired (see Pugh, 1990: 90; Beyer, 2009: 254).<sup>15</sup> A range of factors could be responsible for this and similar experiences. First, there was no scientific or standard way of doing these calculations. Each company (or each division in the case of large corporations) conducted their business in its own way. Second, given that the first business customers of organizations were large corporations, very few individuals could claim that they understood the totality of even the most basic procedures (see Haigh, 2001). In fact, in an advance publicity/opinion piece in *Harvard Business Review* of August 1954, the manager in charge of the acquisition mentioned above had written (Osborn, 1954: 106):

“We were fortunate in already having in our Business Procedures Section at Louisville a group of people who had had training and experience in previous systems and procedures, who were familiar with the operation of our business, and who had at least a college diploma and in many cases a Master in Business Administration degree. (If our initial applications had included mathematical or advanced engineering problems, it would of course have been appropriate to include representatives from those groups too.)”

Thus, what was essential was a correct understanding of, for example, accounting methods and procedures and more generally, business operations of the company and an ability to articulate those procedures in a way that could be translated into computer algorithms easily and without errors. It was no surprise therefore that many companies chose to train their own employees to become programmers rather than hiring individuals from outside. One 1955 survey reported that many user organizations had found it “much easier to teach our personnel to program than to teach outside experienced programmers the details of our business” (quoted in Baum, 1981: 48).

It is clear that, in this context, advanced mathematical skills were of little value. For one thing, mathematical skills required for understanding business calculations (such as accounting and payroll) were trivial. For another, even without the advance of automatic programming languages, programmers could not proceed to lower levels of programming (where advanced mathematical skills would be useful) before a clear formulation of problem. But what skills could enable programmers to understand problems in the business context and did those skills belong to programmers’ work or were they part of an entirely different job. The answer was not clear. It is not surprising therefore, that although there was a recognition of the problem of skills very early on, there was no agreed solution.

---

<sup>15</sup> Gray, G (2001) ‘UNIVAC 1: The First Mass-Produced Computer’, *Unisys History Newsletter*, 5 (1).

# *Business Methods* **ANALYSTS**

To fill the rapidly growing needs of business and industry for advanced forms of computers and data processing machines, IBM is greatly expanding its Advanced Engineering systems planning group. Excellent opportunities exist for men with backgrounds in methods analysis, business administration, or industrial engineering to join this group as Operations Analysts.

Work involves the study of different business operations and formulation of detailed statements of the steps in carrying out the operation. Advancement opportunities exist in Operations Research, Systems Planning and Programming for those who demonstrate ability.

For further details, write to G. W. Woodson, Research Laboratory, International Business Machines Corp., Poughkeepsie, New York.

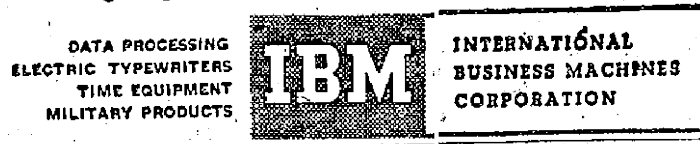


Figure 5-2: IBM advertisement for Business Analysts, 1956.

Given the lack of a common understanding of the problem and the expected rapid rise in the number of computers, people (particularly from the industry) began to raise the issue. In 1953, J.C. McPherson, IBM's director of engineering, attended the Conference on Training in Applied Mathematics and told participants that 1500 mathematicians were imminently needed to match the growth in the number of computers in the businesses or government (see Fay, 1953: 90). In 1954, the first Conference on Training Personnel for the Computing Machine Field was held at the Wayne State University to discuss the problem of supply of computer workers. As Avrid Jacobson, the conference chair, put in his opening remarks: "With the new applications of electronic techniques in accounting and business management during the past year, it is timely that the problems of training and manpower should be examined." (Jacobson, 1955: 3).

Participants from the industry, government and education gathered to exchange views on the nature of demand as well as its quantity. As a GM representative put it, there was "a universal feeling" about "a definite shortage" of people with adequate skills (p.16). Some (mainly from scientific institutions and government) viewed the issue entirely as a matter of mathematical skills and methods and tended to focus on the need for more mathematicians. This included a group who were aware of the potential growth of computer use in business

or “non-scientific areas of use”. For instance, Lt. Col. C. R. Clegg (from the Air Material Command) said “the real quantity of requirements for personnel, with the possible exception of maintenance technicians and electronic engineers, will be in [the non-scientific] area...” but added “We cannot ignore -even for a moment forget- that sophisticated mathematical techniques have a very definite place in the nonscientific as wells as in the scientific applications” (p.11). Others took a broader perspective and emphasized the need for interdisciplinary collaboration in training as well as research (Iverson, 1955).

There were also estimates about the extent of present and future demands, although there was little certainty or agreement over the numbers. Herbert Grosch (from GE, and an ardent opponent of programming languages) estimated the number of current jobs to be somewhere between 2000 and 4000. He expected that the number of positions would double every year, reaching one million jobs by the end of 1960s. In contrast, projection based on a survey of largest 500 manufacturing companies<sup>16</sup> by the Burroughs Corporation had suggested the number of personnel needed for all 500 to be 2325. But as things stood, the survey had found that only 27 out of 139 respondents (response rate 28%) were already using computers or had them on order. Another 23 companies were considering their feasibility and 6 had found them uneconomical. In terms of application of computers, 58 percent were used for engineering calculation, 26 percent for scientific or research purposes and 16 percent for business applications. A significantly high proportion of companies that had computers on order were going to use them in business applications, suggesting a strong trend towards those applications. All computer users had also submitted their staff numbers (See table Table 5-1).

	‘Analyzers’	Programmers	Engineers	Operators	Technicians	Others	Total
Percentage reported	43	17	8	25	3	4	100
Estimated number of employees	1000	400	175	590	70	90	2335

Table 5-1: Distribution of different kinds of workers that were needed for operation of computers based on a study presented in the 1954 conference. Estimates of the number of employees given are based on a linear projection of reported worker requirements to all 500 companies.

<sup>16</sup> Retailers, utilities, banks and insurance companies were excluded because they were believed to have special computing needs.

## **COMPUTER METHODS ANALYSTS**

Honeywell's DATAmatic Division has just introduced the fully transistorized

### **HONEYWELL 800**

The HONEYWELL 800 solves both SCIENTIFIC and DATA PROCESSING problems more efficiently than systems designed specifically for either purpose. In fact, through a unique feature of automatically controlled parallel programming and unsurpassed central processor capabilities, The HONEYWELL 800 can handle BOTH types of problems SIMULTANEOUSLY.

The introduction of the HONEYWELL 800 has resulted in IMMEDIATE openings for Methods Analysts, both in our Main Office in suburban Boston and in our Branch Sales Offices located in key metropolitan cities across the country. These positions involve study of customer's business and scientific problems, preparation of proposals and assistance to customers in the continuing effective use of equipment.

If you have the following qualifications, consider these challenging opportunities to participate in DATAmatic's continuing growth in the electronic data processing field.

- Computer programming experience.
- A good familiarity with office and accounting methods, systems and procedures, or scientific computation.
- A Master's or Bachelor's degree with high scholastic standing.

*For interview consideration, send complete resume in confidence, outlining interests, experience, education, and other information to: John H. Koenig, Professional Employment Supervisor, DATAmatic Division of Minneapolis-Honeywell Regulator Company, 151 Needham Street, Newton Highlands 61, Massachusetts. All inquiries will be answered promptly.*

# **Honeywell**



**DATAmatic**  
ELECTRONIC DATA PROCESSING

Figure 5-3: Honeywell's ad for 'Computer method analysts', 1959.

On the other hand, there was little preparation for this growth in the industry or academy. According to the survey findings, 37 percent of computer personnel in the companies that used computers were recruited from colleges or universities. Another 37 percent were trained in company training programs. 12 percent were provided by vendors while advertisements and government agencies each supplied seven percent of the personnel. Moreover, 14 out of the 27 companies reported that they had difficulty in filling their positions. In-house training was more an act of last resort rather than preference. It was generally considered as a time-consuming and expensive program with unsatisfactory outcomes. Vendor training was usually effective but it had a very small capacity, Manuel

Paul Chinitz<sup>17</sup>, the director of training at Remington Rand, confessed that in the 15 months since the establishment of their center in New York city, they had trained only 162 programmers. In his estimation, the total training capacity of all manufacturers combined was around 260 programmer per year. The situation was hardly better at universities. P. E. Little of Wayne University (a pioneer of providing computer training at the time), described the situation as this:

“Less than twenty institutions of higher learning have formal courses directly related to large automatic computing machines. Less than half of these have a large machine available for the use of teachers and students, and few of these machines are being used in the study of problems related to the data processing and management decision requirements of business.”

The message from conference participants was clear: a “cooperative effort” had to be put in place to rescue the field of computing from collapse.<sup>18</sup> The “persistent personnel problem” became the main theme of conference over the next several years (Ensmenger, 2010a: 59).

It seems that at least at one point in time, not only experienced programmers were hard to find or train, but also it was difficult to find people who would be interested in taking up a programming job. An interview published in the “Talk of The Town” column of the *New Yorker* magazine in January 1957 can illustrate this point. After seeing job advertisements looking for “research programmers for digital computers” in the *New York Times*, the reporters had contacted IBM to ask about this new ‘profession’ which they had “never heard of”. They also wanted to know if anybody had applied. The IBM manager in charge of recruiting programmers, Robert W. Bemer, had told them that the ad (which was also run in the *Los Angeles Times* and the *Scientific American*) “had brought a total of seven responses”. The reporters had found this less surprising than the fact that IBM considered this “excellent” result.

Bemer (a programmer himself)<sup>19</sup> had explained to them that programmers were “the clever fellows who figure out the proper form for stating whatever problem a machine is expected to solve.” In his estimation there were nearly 15000 programmers in the U.S., but there were

---

<sup>17</sup> Chinitz was a mathematician who had worked in U.S. Navy Computing Labs before joining the Eckert-Mauchly Computer Corporation in 1950.

<sup>18</sup> The federal government and its agencies (the Air Force, Army, Navy, Atomic Energy Commission, etc.) each had a small training laboratory dedicated to meeting their own staffing needs. The Army base at Aberdeen Proving Ground, for instance, preferred to hire people with at least a Master’s degree in mathematics (though occasionally, it hired and trained people with a bachelor’s degree too). In 1953, it employed a total of 70 people.

<sup>19</sup> Bemer, himself had joined IBM in December 1955. With a B.A. in mathematics and a certificate in Aeronautical Engineering, he had first worked as programmer in Rand Corp in 1949.



# RESEARCH PROGRAMMERS *for* DIGITAL COMPUTERS

**IBM** is looking for experienced and ingenious designers of computer programming systems to staff an expanding research effort in the development and automatic translation of a multi-computer language. Current and proposed studies in this field offer a challenge and opportunity unequalled in the computing profession. *Rewards are commensurate with the high qualifications required.*

Those who qualify may have helped to develop or use advanced programming systems written in abstract or synthetic languages. In addition to the prime requisite of varied computer experience, knowledge in a related field such as language theory, logic or topology is welcome. Those who enjoy playing chess or solving puzzles will find this work absorbing.

If you are interested in joining the most advanced and vital programming effort in the computer industry, and possess a college degree or equivalent experience, write to:

R. W. Emer, Dept. 731A, Programming Research  
International Business Machines Corporation  
590 Madison Avenue, New York 22, N. Y.

*Be sure to include a resume of your experience.*

DATA PROCESSING  
ELECTRIC TYPEWRITERS  
TIME EQUIPMENT  
MILITARY PRODUCTS



Figure 5-4: IBM's programmer ad, 4 Nov. 1956

1500 computers installed with "larger models" requiring a team of between 30 to 50 programmers, each. So, there was a huge gap between demand and supply of programmers and naturally, most existing programmers were paid very well. Since they were unlikely to be "mooning over 'Help Wanted' ads", IBM had placed a 'display ad' (rather than a conventional classified ad) to lure them. Of the seven responses he had received, five were from experienced programmers and two from interested individuals with no experience.

The fact that in the "few weeks" since the appearance of the ad only two 'interested' individuals had applied suggests that the moment of take-off for programming jobs had not yet arrived. Part of the lack of interest on the part of general public can be explained by the fact that few people knew what programming (let alone 'research programming') involved. But the ad too did not clarify what the job was about. The confusing requirements that the ad expected the candidates to have reflects very well the confusions about skills which was

discussed above: After stating that they are “looking for experienced and ingenious designers of computer programming systems”, the ad went on to say: “*Rewards are commensurate with the high qualifications required.*” (Original emphasis). Then those who will be qualified were described as having experience in developing or using “advanced programming systems written in abstract or synthetic languages” (this being a “prime requisite”). Knowledge in “related fields” (encompassing “language theory, logic or topology”) was considered an advantage. But then the tone of ad changes to suggest: “Those who enjoy playing chess, or solving puzzles will find this work absorbing”. It concluded by inviting those who are “interested” and have “a college degree or equivalent” to contact Bemer at IBM “Programming Research” Department.

Evidence for the confusing effect of the ad can be found in Bemer’s remark that one of the two unexperienced individuals “really was interested only in playing chess” and was sent back to “his board”. But at the same time, Bemer’s reaction shows that IBM was generally open to people without qualification provided that they had “the kind of mind [they] like”. Eventually, the New Yorker story attracted a wider audience and more candidates applied.

Recollections from one of the successful applicants who had become aware of the IBM recruitment program thanks to the New Yorker piece confirm this sense of confusion as well as urgency inside IBM. Mark Halpern (at the time an English PhD candidate) recalled his approach to the company as below:

“[After reading Bemer’s remark, I wrote] a letter in which I told him that I was very smart and asked for an interview. Such was IBM’s need for programmers, and so slight was anyone’s knowledge of what made for a good programmer, that my letter was answered promptly with an invitation to come down for tests and interviews.”  
(Halpern, 1991: 102)

Soon he found himself employed as a programming trainee, working along veterans and “new kids” like himself. The diversity of people who he met as colleagues in the first week of work is interesting: a crystallographer, a farmer, a former “high-fashion model”, a writer/musician, and several avid chess players. This situation, which was by no means exceptional, shows how little choice computer vendors initially had when ‘selecting’ their intake from applicants.

## **5.5. Analysis**

From the discussion above, it seems clear that the advance of automated programming languages was not enough for making mathematical skills irrelevant in the labor market. If

the use of computer was confined to scientific and engineering applications, programmers still had to have mathematical skills. This is evident in the fact that in the mentioned application domains, mathematical skills are still necessary and highly valued. What was really important for the relative decline of value of those skills in the labor market was the fact that those skills had little contribution to the new and emerging contexts of computer use for business purposes. In 1958, a report for the Bureau of Labor Statistics had found that “many employers no longer stress a strong background in mathematics for programming of business and other mass data if candidates can demonstrate an aptitude for the work. However, programmers of scientific and engineering problems are usually college graduates with a major in one of the sciences or in engineering and some course work in mathematics” (Paschell, 1958: 11).<sup>20</sup>

At the same time, such efforts toward automation never entirely removed either the dull and repetitive aspects of programming or the possibility of making errors. It merely moved them to a different level (Ensmenger, 2010a: 44, 109). Programmers still needed to write code, test, debug, etc. However, by automating part of the programmers’ job, programming languages made wider use of computer technology possible and hence, contributed to growth of demand for programmers. In fact, there is no evidence that automated programming negatively affected computer workers in terms of job market and organizational position. The absolute growth of the number of programmers in the 50s and 60s meant that even coders whose jobs were eliminated by these developments were most probably able to find new employment quite easily. So perhaps, William Aspray is only slightly exaggerating the effects of programming languages when he says (2004: 90): “They did not make any given programming task less time-consuming for the human programmer, but they just opened up the flood gates to doing more programming.”

This is not to say however, that the advance of automated programming languages did not affect the direction of changes in IT work. Programming languages were pioneering developments of layering the technical object of software work. The widespread use of programming languages raised the platform of software work by building one level of auto-

---

<sup>20</sup> At about the same time, researchers reviewing IBM’s Programmers Aptitude Test reported with a sense of surprise that in their study, “majoring in mathematics was not found to be significantly related to performance as a programmer!” (McNamara and Hughes, 1961: 41-2). As McNamara explained later, “a careful analysis of the situation [had] indicated that mathematical knowledge was associated with the application and not with programming ability” (McNamara, 1967: 53). For more discussions on aptitude tests see the next two sections.

# COMPUTER PROGRAMMERS

Opportunities high in professional challenge  
are now open to you with General Electric at the  
Huntsville Computer Center, Huntsville, Alabama.

## ■ *Scientific Programmers—B.S., M.S., Ph.D.*

Every launching of a missile or satellite by the Development Operations Division of the Army Ballistic Missile Agency—directed by Dr. Wernher von Braun—is supported by weather, flutter, and fuel flow analysis and flight predictions calculated by SCIENTIFIC COMPUTER PROGRAMMERS. And this is only one phase of the broad computer program operated by General Electric for the Computation Laboratory of ABMA.

Current openings necessitate minimum of one year's experience in programming for large-scale scientific and business data processing machines.

## ■ *Business Programmers—B.S., B.A., M.A.*

A corps of business programmers here is concerned with management analyses of an extremely complex nature, covering logistics and inventory control—costing—funding and other aspects of large-scale weapons system projects. There are a limited number of immediate openings.

## □ *Also, openings for people with Computer Techniques experience.*

Compensation is commensurate with academic training and computer experience.

Excellent employee benefits, including insurance, pension, vacation and stock purchase programs.

Equipment Available:  
709, 704's, 705,  
Datatron 205's

Huntsville, Alabama, is a cosmopolitan community of 70,000 with world-renowned professional people. The Computer Center is in the heart of TVA's lake and mountain playground, offering year-round outdoor recreation.

If interested, write in confidence to: Mr. B. G. Cole, Dept. 111-B

**HUNTSVILLE COMPUTER CENTER**

**GENERAL  ELECTRIC**

Box 988

Huntsville, Alabama

Figure 5-5: General Electric's ad for programmers. Notice the distinction between scientific and business programming. 1959

mation upon another level of automation. This layering shifted the focus of technical efforts and discussions to another level without resolving previous issues. Interestingly, as early as 1954, in the first Symposium on Automatic Programming for Digital Computers, one participant suggested that:

“universities were reluctant to train programmers, feeling that there are too many specialized codes. Perhaps this reluctance will vanish if we can provide a code more or less independent of the machine.” (Gorn, 1954: 75)

But the overall focus of industry was towards developing programming languages as replacements for programmers. Though this objective never materialized, it was an

important factor in shaping the direction of industry and the discussions that took place about the required skills.

In the early 1960s, when “programming languages” were accepted as the way forward, the unanswered questions about skills were being reformulated. Some envisioned that “common programming languages” would provide a solid foundation for defining both the role of programmers and their required skills. For instance, in 1962 an elite group of programmers were gathered in the Fifth RAND Computer Symposium to discuss the “Pros and Cons of Common Languages”. High on the agenda was the question (p. 5):

“How are [we] going to train people? What kinds of people are going to use a common language? (We're talking here about the level of the users. Are they going to be top-notch programmers, or are they going to be clerks?)”

During the discussions, when one participant suggested that:

“Maybe they should consider standardization of programmer levels. Maybe we should define what a guy ought to know before he calls himself a programmer. Maybe if we establish certain minimum standards of programmer competence we might then have a lot less trouble with worrying about standards of programming language and object code efficiency.” (p. 122-3)

Another participant suggested that this was not possible: “There is too much of a shortage of people right now to enforce standards like that” (see also p. 153-4 and 162-3). This brief exchange was quickly drawn back into a discussion about the relative advantages and disadvantages of different languages and their associated technics (see also p. 153-4 and 162-3), but it serves to show how ‘old’ problems resurfaced to shape the discussions around skills in the ‘new’ context.

In order to understand the role of programming languages in the formation of skills, one must place them in their context of development and use. Different languages were suitable for different applications in different contexts. By 1991, at least 1000 programming languages had been developed in the U.S. alone. However, very few of them were used for more than a short period of time or beyond a fairly specialized application domain (see Sammet, 1991). At the same time, as Friedman and Cornford points out, some applications were still written in machine code (Friedman and Cornford, 1989: 117). According to them, this was happened the value of various tools (and trainings) for computer workers was “contingent upon the programming environment in which they were introduced” (p. 251). The centrality of concerns about the costs of programming led Friedman and Cornford, among others, to believe that the development and use of programming languages (and

similar tools) followed a simple rational-economic calculation (see Friedman and Cornford, 1989: 77-81, 93-94 and 116-120).

This is a plausible argument because, in the late 1960s and early 1970s, the industry was deeply concerned about the rising costs of software development. A series of articles published in a variety of outlets all pointed to the imminently rising share of software in the development of computer systems (most famously, Boehm, 1973). As subsequent historical analysis has shown, however, these analyses followed the appearance of a diagram by Informatics co-founder, Werner Frank, which was published in June 1968 in *Datamation*.<sup>21</sup> Nearly all of these articles reproduced a similar diagram based on speculation rather than actual data. Moreover, later studies have shown that the ratio of software to hardware costs has remained “more or less constant” (Ceruzzi, 2003: 82).

So it is important to attend to the factors that shaped the perceptions and expectations of the industry. I have already mentioned the initial resistance that Hopper faced in trying to develop early programming languages. Backus in IBM encountered a similar situation when he wanted to initiate the process though he faced less resistance (partly thanks to Hopper and the competition between IBM and Remington Rand, see Bashe et al., 1986: 339-40).<sup>22</sup> But once established as a valid tool for programming, there was almost no limit to their proliferation. Knuth and Pardo (1977/2002: 77) point out that an “explosive growth in language development” occurred after 1957. Nevertheless, the kind of argument that Hopper made to convince her managers was later used as the ‘sales pitch’ of the computer industry to sell programming languages to customers. Programming languages were heavily promoted as the tools that would make programming a straightforward and error-free task and eliminate the need for programmers. In the brief exchange at the RAND symposium reported above, one of the participants expressed his frustration over the promises of reduced (if not eliminated) need for programmers that accompanied programming languages:

“[...] I’ve never seen a hot dog language come out yet in the last 14 years — beginning with Mrs. Hopper’s A-0 compiler ... that didn’t have tied to it the claim in its brochure that this one will eliminate all programmers. The last one we got was just three days ago from General Electric [...] Managers can now do their own programming; engineers can do their own programming, etc. As always, the claim seems to be made that programmers are not needed anymore.” (p. 123-4)

---

<sup>21</sup> Frank himself, later called it a “software myth”. Campbell-Kelly (2003: 91-95) provides a critical review of this “impressionistic” diagram and its derivatives.

<sup>22</sup> The case of COBOL is slightly different because it was based on a government initiative but it built on the work of Hopper, Backus and others and was driven by the same cost-saving reasoning.

Another participant pointed out that many such advertisements were published in *Time*, *Business Week* or *Wall Street Journal* rather than *Datamation* or *Communications of ACM*. In the words of another participant “They were talking to the guys who buy the machines” (p. 124). As I explain in the next section, computer manufacturers actively sought and created a market for their products. The main concern of most of sales representatives (especially at IBM) was selling the hardware (Grad: 2002, 68) and therefore, they tended to exaggerate the ease of programming.

It is clear from the evidence there was a significant amount of push by the vendors to shape the decisions of managers in user organizations. While the power structure underlying managerial decision-making inside organizations (e.g. technological development and resource allocation) is well known and often taken-for-granted, the wider sources of influence (e.g. King et al., 1994) are less well understood.<sup>23</sup> For instance, FORTRAN and COBOL, two of the most successful programming languages that had the most lasting effect on the formation of skills, clearly benefited from the dominant position of their supporters. In the case of FORTRAN, which was developed at IBM, its adoption was facilitated by the fact that IBM had helped to establish computer laboratories in universities across the United States. Apart from donated computers, IBM offered up to 60 per cent discount to the universities which would provide courses in “business data processing” and “scientific computing” (Watson and Petre, 1990: 261; see also Brock, 1975: 156-8). COBOL, on the other hand, which was based on a collaborative initiative by computer manufacturers, users and academics, did not enjoy widespread acceptance in the industry standard until the U.S. Department of Defense made it the standard for developing defense-related software. In contrast to FORTRAN, which was received well, the success of COBOL has been despite its technical inadequacies (see Ensmenger: 2010a, 93-101).

Thus, we can see that once programming languages became part of the toolkit of programmers, skills associated with them became part of the skill profile of what it meant to be a programmer. But these developments were happening in a social context and actors other than programmers had an interest in the way those skills were valued or evaluated. IT corporations, for their part, pursued two strategies: on the one hand, they tried to convince their customers that their new computers (or development tools) eradicated the need for sophisticated skills of programmers and hence would lower the costs of programming. On the other hand, competition over computers was extended to competition over programming

---

<sup>23</sup> King et al. (1994) use institutions in a sense that is closer to what I have considered as actors in this study. Their examples of “institutions influencing IT innovation” are “government authorities”, professional and trade and industry associations”, “trend-setting” and “multinational” corporations, etc. This is why I have referred to them as sources of influence rather than “institutional factors”.

languages and each company tried to make their programming tools as widely used as possible (the case of IBM and Honeywell resisting COBOL as the industry standard is one example, see Fisher, et al., 1984: 71; Ensmenger, 2010a: 99-100; see also Bemmer, 1969 and Misa, 2012: 55-6 for discussions about ALGOL). Initially these efforts were only part of an effort to enhance the sale of computers (programming languages were not sold). But as the competition became more complex and these tools became sellable pieces of software (like operating systems and CASE tools),<sup>24</sup> they become profitable items in the competition in their own right.

Inevitably, evolution of the competition tied the value of certain skills to the value of specific computers or programming tools, becoming part of the dynamics of formation of skills in the industry. As such, the training of software workers and diffusion of products in the product market became inter-related elements in the competitive strategies of IT corporations. They could, and often did, use one of these elements to affect the other (like IBM, above). While not all of these efforts were equally effective, I will discuss those efforts that had greatest impact on the formation of skills in the next chapters. As I will try to show, soon the problem for employers was not the shortage of skilled workers as such, but the shortage of workers with ‘appropriate skills’.

So what we see in this stage is the development of a regime of skill formation that is influenced to a large extent by the competition between IT corporations. At the same time, as I will show in the subsequent chapters, the expansion of computer use meant that job opportunities were so abundant that computer workers were not alarmed by almost any challenge for their working practices or change in skills. In fact they seem to have been delighted by the choices. If they were really unhappy with their situation or position they could simply quit and find another employer. In the absence of any collective action by software workers, the only other actor who could intervene (in the competition or, more directly, in the shaping of formation of skills) was the U.S. government. While the government had made investments that were critical in the early stages of industry (Flamm, 1988), it did not attempt to regulate the labor market or directly influence the formation of

---

<sup>24</sup> I consider operating systems a programming tool because of their effect on the reduced need for re-programming. Many operating systems were initially called “operating languages” while an early programming textbook referred to the use of programming languages as “machine-aided coding” (see McCracken, Weiss and Lee, 1966). Operating systems automated certain repetitive tasks that were routinely required when running programs with hardware (see Friedman and Cornford, 1989: 74-7 and 116-17). As we will see, later operating systems saved programming efforts by providing a consistent platform for programming across hardware generations (beyond several runs of a program). They also automated part of the activities that was delegated to computer operators (see Ensmenger, 2009). For a discussion of controversies about the changes in skills which accompanied the development and use of CASE tools, see Orlikowski (1989).



skills and its main intervention in the competition did not occur until 1969. Nevertheless, when the SAGE project, which started in the early 1950s, required a large number of programmers, a non-profit corporation was assigned and funded by the project to both train programmers and perform the programming tasks that were required for the project. Development of this corporation, which was formed largely outside the competition, significantly contributed to both expanding the number of available programmers as well as shaping the conditions of entry to the labor market. Therefore, in the next section I will relate the history of this corporation and its long-standing effects.

**“MY NEW BUSINESS PARTNER IS A MANAGEMENT GENIUS, EASY TO TALK TO, AND DOESN'T COST ME MUCH.”**

and receivables—the cash flow and management data you need to make your business more efficient and competitive. Another demonstration of the big idea at Sperry: Making machines do more, so man can do more. Sperry Univac and Sperry Vickers are divisions of Sperry Rand Corporation, 1250 Avenue of the Americas, New York, New York 10020.

**Sperry Univac's new BC/7 computer is a perfect partner for any small businessman. It's compact, inexpensive, and it speaks and understands English.**

Making computer technology easy for a small business is a challenge. Sperry Univac's new BC/7 computer meets that challenge. It fits into a small operation with minimum fuss. Like a good business partner. The BC/7 speaks and understands English, so you won't need a professional programmer. A few days' training and you'll be completely at home with it. It can cost less than \$1,000 a month, and for just, well, a few dollars more.

It doesn't need a big office, or special air conditioning. It even has its own health plan. If anything goes wrong, you call just one place: Sperry Univac. The BC/7 gives you more facts about your business faster than you ever got them before: order entry, inventory, payroll, production, payables

**SPERRY**

**Making machines do more so man can do more.**

Figure 5-6: Sperry Univac ad for new computer, New Yorker, 2 Oct. 1978

Excerpt from the text: “The BC/7 Speaks and Understands English, so you won't need a professional programmer. A few days' training and you'll be completely at home with it. It can cost less than a \$1,000 a month...it's sheer genius.”

## 5.6. The Origins and Legacy of SAGE Project

In the early 1950s, the U.S. government was not particularly alarmed about the shortage of computer workers. On the one hand, there had been an omnipresent sense of shortage of “scientific and professional manpower” across the U.S. industries since at least the start of the decade<sup>25</sup> (NSF, 1955). On the other hand, computer workers were still a tiny section of mathematical occupations, with no significant presence even at government organizations. The 1954 conference was, in part, an attempt to involve government but it was to no avail (see Aspray, 2004). The biggest help for the computing industry, in terms of training of programmers, came in the form of a division (later, a spin-off) of the RAND Corporation which was responsible for developing the ‘system’ (i.e. software) for the computers of the Semi-Automatic Ground Environment (SAGE) missile defense project. As we will see, many of the consequences of this development was unintended and unforeseen and went well beyond the quantitative supply of programmers in the industry.

Between the years 1949 and 1952, alarmed by the possibility of an air-borne Soviet attack on the U.S. Soil, the US Airforce initiated the SAGE project to speed up and inter-relate the processing of data produced by its distributed radar system. At its core, the SAGE computer system was based on Whirlwind, the first real-time digital computer, which was developed by a group of MIT scientists and engineers initially for the Navy but later for the Air Force. The production of an enhanced reliable version of Whirlwind on an industrial scale (for one national and 24 regional centers) had to be contracted to a manufacturing company with adequate resources. The MIT team members approached several manufacturing companies. After having discussions with each company and evaluating their capabilities, the MIT team selected IBM in 1952. MIT team members considered a comprehensive list of factors in the decision (see Astrahan and Jacobs, 1983: 344) but, as Ceruzzi (2003: 362) suggests, it seems that ultimately “the orderly production facilities” of IBM had the biggest impact on the decision (see also Watson and Petre, 1990: 246). According to Cuthbert Hurd, IBM’s director of the applied sciences at the time, the decision was based “on [the] assembly line kind of concept for quantity production and [on] the quality of [IBM’s] people” (quoted in Fisher et al. 1984: 27).<sup>26</sup> Between 1952 and 1954, when IBM was finally awarded the contract, IBM and MIT engineers worked on increasingly detailed specifications of the system.

---

<sup>25</sup> Katzin, L.I. (1952) Scientific Manpower, *Bulletin of the Atomic Scientists*, 8(1) p.27-31; Katzin, L.I. (1953) Scientific and Professional Manpower. *Bulletin of the Atomic Scientists*, 1953 9 (6), 225-6

<sup>26</sup> Several other contractors were selected for providing other components (see Baum 1981: 25, Jacobs, 1983).

SMALL BUT NATIONALLY KNOWN MANUFACTURER OF PRECISION MOULDED INDUSTRIAL RUBBER PRODUCTS REQUIRES A MANUFACTURING MANAGER AS PART OF ITS PROGRAM OF REORGANIZATION. DUTIES INCLUDE DIRECTION OF WORKING FORCES (APPROXIMATELY 500) AND APPROPRIATE STAFF FUNCTIONS. CANDIDATE WILL REQUIRE ENGINEERING DEGREE AND PROVEN RECORD AS SUCCESSFUL MANAGER OF A SMALL TO MEDIUM SIZED OPERATION IN THIS FIELD LOCATION, PHILA. AREA. SUBMIT FULL RESUME AND SALARY REQUIREMENTS TO G-16, P. O. BOX 3552, PHILA. 22, PA.

MASSEURS. Husky No exp nec Send snapshot, non-return. B-24 Inquiret

Mathematicians  
Actuarial Students  
Math Analysts  
Computer  
Programmers

Outstanding opportunities in the field of scientific computing.

THE  
Rand Corp.  
Santa Monica, Calif.

Work in Rand Corp.'s large digital and analog computer installations. Excellent working conditions and employee benefits.

Requirements:  
Bachelor's degree or higher.  
Experience desirable but not

Figure 5-7: Rand's programmer ad, 1954.

IBM found the task of programming for SAGE too difficult (Ensmenger, 2010a: 60). Robert Crago, the engineer in charge of IBM's Engineering Design Office, later recalled that they had estimated that "several thousand" programmers were needed to finish the project. Bell Telephone Laboratories also turned it down (see Tropp, 1983: 385-6). The MIT team, which was responsible for the overall design of the system and its software ("master computer programs"), looked for an alternative company who would be responsible for "operational development" of those programs as well as creating the supporting software, installing and adapting programs to each site and making updates (Baum, 1981: 22). In 1955, RAND Corporation, a nonprofit think-tank based in Santa Monica (CA), was selected without competition. The choice was primarily based on three points: 1) as a military-affiliated research organization, RAND needed no security clearance. 2) RAND employed a relatively

large number of experienced computer programmers (twenty-five, to be exact).<sup>27</sup> 3) Since 1950, RAND employees had been involved in programming for air-defense problems like flight simulation (see Baum, 1981: 22-3).

In July 1955, a team of five RAND programmers joined the MIT team at the Lincoln Laboratory, Lexington (MA) to initiate the programming effort. It seems that MIT-RAND team had massively underestimated the number of the programmers that were going to be needed (Baum: 1981: 35; see also Jacobs' comments in Tropp, 1983: 386). Initially, RAND was expected to provide "a total of seventy programmers" for the design of the system (Baum, 1981: 23). By September a new division, called System Development Division (SDD) was established for the purpose of concentrating computer-related activities (system development, computer programming and system training)<sup>28</sup> in RAND. The target of 70 programmers was passed by December 1955 but there was no end to hiring in sight. SDD continued to hire at an average rate of 50 employees per month and by 1956, it was employing 1000 people. While not all of the new recruits were programmers (or computer-related workers), a significant proportion was. A note published in March 1956 in "The RANDom News", the internal magazine of RAND, underscored the rapid growth of SDD and advertised nearly 150 new jobs that were available. Among them were positions for 70 programmers, 14 tabulating operators and 3 keypunch operators. Over the following months, long lists of "new faces" filled the pages of every issue of The RANDom News.

A glance at the qualifications of the recruits confirms the previous observations about the variety of people who were admitted. At the end of the list of 'new faces' that were receiving programming training, the May 1956 edition of the magazine simply commented "...varied backgrounds, to say the least". Notable backgrounds on the series of lists published throughout 1956 included: a furniture salesman, a real estate salesman, an educational-psychology major, a literature major with surveying experience, a "wheat rancher" and a "former buyer for a department store". Clearly, most of the newly recruited employees had no previous experience or training in programming. Initially, IBM provided training at the plant which it had specifically set up for the SAGE project in Kingston (NY). From January 1957, IBM's eight-week course was augmented by another eight weeks of

---

<sup>27</sup> Baum, the historian of SDC, quotes SDC's first president as claiming that total number of "talented programmers" in the U.S. at the time was 250. Baum also refers to an unnamed "industry bible" which lists 1200 programmers in 1955.

<sup>28</sup> System training, in this context, referred to training and preparation of radar personnel at Air Force bases with simulated air raids and similar situations. "System" and "System development" (as in the division title) referred to the design (and development) of the software to be used in such tasks as well as those related to the SAGE project.

advanced training by SDD. By mid-1957, SDD became solely responsible for the full course of programmer training, offering it in both Lexington and Santa Monica (Baum, 1981: 50).

*Heralding*

## A NEW PROFESSION!

A College Math or Science Background  
Can Start You on a New, Highly Specialized  
Career with a Challenging Future

### DIGITAL COMPUTER PROGRAMMING

If your present position no longer offers you a sufficient challenge—if you would like a ground floor opportunity for a new career in a stimulating new field with a new organization, then it will be to your advantage to investigate what RAND has to offer.

RAND COMPUTER PROGRAMMERS are major contributors in the establishment of the SAGE continental air-defense network—the largest and most complex automated system yet devised, employing the most advanced digital computers in existence. What's more, RAND is charged by the U. S. Air Defense Command with the continuing responsibility for the development and advancement of SAGE. Such sweeping potentialities offer long-range careers in the very forefront of a challenging new profession.

A PROFESSIONAL PROGRAMMER works out ways of using high-speed computers to solve complex real-time problems. This involves a comprehensive analysis of the problem, formulation of the logic to be used in the solution, coding it into the

computer's language and checking the completed program—a total operation that may take weeks of mathematical reasoning to arrive at the most efficient solution.

TO QUALIFY FOR TRAINING AS A PROFESSIONAL DIGITAL COMPUTER PROGRAMMER (in Lexington, Mass.), you must have college mathematics through calculus, a high aptitude for logical reasoning, U. S. citizenship, and the willingness to relocate. Computer experience is unnecessary. Salary potential and company benefits are on a professional level.

TO ARRANGE for a confidential interview this week in which a firm commitment may be reached, telephone collect:

**Robert Frost**

Cal Monday, Tuesday or Wednesday,  
May 20, 21 or 22

PHONE: ELdorado 5-2600

Hours: 10 A. M. to 1 P. M. and  
2 P. M. to 7 P. M.

If you cannot call, write for employment application and literature to individual named above at the address below.

EXPERIENCED PROGRAMMERS ARE ALSO INVITED TO MAKE INQUIRIES

## SYSTEM DEVELOPMENT DIVISION

### The RAND Corporation

Dept. NYT, 2500 Colorado Avenue, Santa Monica, California

(An independent non-profit research organization — long-range scientific advisors to the Air Force.)

Figure 5-8: SDC's programming ad, 1957

SDC continued to grow rapidly after its formal incorporation in December 1957. By 1959, when SDC employed near 2000 employees, more than 800 programmers were working on the SAGE project (Baum: 1981: 35). By 1963, when SAGE system was fully deployed, the number of SDC employees had reached 4300 and more than half of them were programmers. Many of these employees were attracted by an almost unbroken series of ads published in numerous newspapers and magazines (and even over the radio) from 1957 through mid 1960s (see p. 48). Some of these ads (especially in the 1950s) were targeted at inexperienced candidates, while others were intended to attract experienced programmers. At the same time, SDC itself was suffering from a very high attrition rate. Only 50 percent

of the programmer trainees remained with the company more than 4 years after their start date (p. 51).<sup>29</sup> So in 1963, there were an estimated 6000 ex-SDC employees working in computer jobs across the industries. These numbers are significant because they show not only that SDC at the time employed more programmers than anyone else, but also that SDC had trained more programmers than anyone else. It was reported that in every programming department in the U.S., one could meet 2 or 3 ex-SDC employees (p. 47). SDC managers certainly felt that SDC had acted like a “university for programmers” (p. 51).

But even before starting its formal operation as SDC, SDD had problems in recruiting and managing programmers. First, in the absence of any skill or qualification-based criteria for selecting potential programmers, managers were looking for a way of selecting those with a higher potential. Second, it was clear that even after passing the 8-week (or 16-week) training course, the newly recruited programmers, on their own, could not produce programs efficiently and reliably. Programming teams had to be organized in a way that would both make them manageable and reduce their reliance on extensive experience and skills. As we will see, the solution that SDC used for the selection problem soon became a standard in the industry, but its solution to the organization problem had a limited impact.

In selecting their intake of potential programmers, SDC managers, like those at IBM, looked for certain mental capabilities. “We looked for logical minds and a math or science background,” claimed Hal Wilson, SDC’s director of Employee Relations. To identify those minds, SDC managers relied on aptitude tests and psychological profiles that were used previously at RAND. The basic assumption of such psychometric tests was that a good performer in any job was born rather than made. Each individual, it was argued, had certain innate characteristics and capacities that made them suitable for specific jobs. Therefore, employers had to look for specific mental capacities and psychological characteristics to see whether or not an individual was suitable for a specific job.

However, nobody knew what psychological characteristics made an individual suitable for programming. As mentioned above, the roles and responsibilities associated with computer jobs were only vaguely defined. Therefore, in order to ‘caliber’ the selection method at the same time that recruitments were being made, existing tests had to be adapted and experimented with (see Baum, 1981: 52). As early as 1952, RAND had developed methods for selecting programmers (Rowan, 1957). Building on the experiences of RAND and MIT

---

<sup>29</sup> It is worth mentioning that, according to a corporate survey, former employees constituted “a substantial proportion” of experienced programmers that were recruited in later years (Baum: 1981: 52).

Lincoln Laboratory, SDC adapted two aptitude tests (ostensibly testing “mental ability” and “temperament schedule”) for the initial screening of potential programmers. Candidates who scored well were invited to interviews, tested again (for desirable psychological characteristics) and, if they had performed well, were briefed on the assignment and made an offer (Rowan, 1958). In the first six months of 1959, for instance, SDC tested 6850 candidates and hired 800 (Baum, 1981: 49).

The SDC approach to the organization of programmers was far more limited in terms of its external impact. In order to reduce reliance on experienced programmers and use novice programmers effectively, SDC organized its programming activities along the lines of a “software factory”. Following the principles of industrial engineering and scientific management, SDC managers rationalized jobs to create a strict division of labor: in the programming ‘production line’, programmers “were organized into skill centers corresponding to the major steps in the development process- requirement analysis, operational design, program design and production, testing, and site adaptation”. A manager planned, coordinated and guided each piece of SAGE programs through these centers (Baum, 1981: 39). Thus, large pieces of programming work were broken down to simple modules that could be assigned to teams of relatively inexperienced programmers.<sup>30</sup> While novice programmers were responsible for either testing or lower levels of programming, experienced programmers were busy specifying the requirements, designing programs and planning and monitoring other programmers. In short, a structured top-down approach was enacted in both the organization of work and the program output (see Benington, 1983: 356-7).

A great amount of effort was taken to produce programming tools that would help novice programmers generate reliable programs, test them, document the process and provide other managerial and analytical information (Benington, 1983: 351).<sup>31</sup> The ideal was that coding would involve only a “simple mechanical translation” (p. 357). In practice, however, novice programmers had to detect problems in the code and revise coding specifications. Overall, it seems that programmers’ jobs were more limited by the specificity of their assignments rather than repetitiveness of tasks. As Baum (1981: 52) reports in his discussion of high turnover among SDC programmers, as early as 1958, “hundreds of programmers working on

---

<sup>30</sup> The programming effort involved half a million lines of code. Only a quarter of this was for the actual air-defense program. The rest belonged to either supporting programs or tools used in the development process. See below.

<sup>31</sup> These were not automated coding tools, at least not “in the conventional sense” (Benington, 1983: 357) because they were seen as “dangerous”. See below.



specialized SAGE subfunctions...found themselves encased in specialized blocks of [a] pyramid with minimal opportunity to move upward or sideways”.

It is important to note that the daily usage of the term “programmer” in SDC, as reflected in its history, was shorthand for all sorts of organizational positions that were involved in programming activities (see, for instance, Baum, 1981: 35, and 38). Along with the pyramidal structure mentioned in the above quotation, at least as early as 1961, there were 5 levels of computer programmer in SDC. From junior levels to the most senior, it included: “computer programmer”, “programming analyst”, “programming analyst - senior”, “computer systems specialist” and “computer systems specialist - senior”. One’s progress through these levels was primarily dependent on the extent of his or her experience, rather than acquisition of specific skills or certain qualifications.

### **5.7. Analysis**

SDC was founded and run (until 1969) as a non-for-profit organization that had income from doing projects for the U.S. Military (especially Air Force). As such it was largely outside the commercial developments and competition that occurred throughout the 1950s and 1960s. At the same time, it was a training center for programmers and a constant source of innovations in software development. Therefore, its headquarters in Santa Monica was an important source of programmers and programming skills for the region and beyond. Through its training program, it provided the early computer industry with a much-needed pool of programmers. Moreover, its pioneering selection methods defined the accepted approach to selecting programming candidates for many years to come. Even its ideal of “software factory” which was not imitated outside SDC can be seen as an exception that exposed the norm of the time and was later rediscovered as a predecessor of efforts to bring the software crisis under control. I briefly discuss each of these two practices, in the sections below.

#### **5.7.1. Psychometric tests**

Abbate (2012: 51) has pointed out that psychometric aptitude tests were based on assumptions that were fundamentally different from those based on some sort of educational or training certificate, because they focused on innate capabilities rather than learned skills. As such they deserve a special consideration in our analysis of developments in IT workers’ skills. Psychometric tests were first advocated by the pioneers of “vocational guidance” movement in the early 1900s and were later promoted by psychologists who advocated a ‘scientific approach’ to personnel selection (see Jacoby, 1985: 101; Kaufman, 2008: 126). During the two World Wars, U.S. military had extensively used these methods in screening

soldiers. Some analysts believe that the two world wars (which heightened concerns about efficiency and waste) had acted as catalysts for adoption of these methods across the industries (Eilbirt, 1959). But the SDC psychologist in charge of administering these tests had emphasized in a paper that (especially after the First World War) army use of these tests had made them “fashionable” and, generally, these tests were “oversold” with “extravagant claims”. He could only express hope that, in 1957 - by which time these tests were used “everywhere and for every purpose”- the situation was “somewhat healthier” (Rowan, 1957: 348).

Some historians suggest that these tests were the only option available to SDC (e.g. Ensmenger, 2010a: 63). This seems to be a correct observation at least in one specific sense: the immediate institutional environment of SDC managers provided little choice. On the one hand, as it was mentioned above, RAND -where use of these methods for selecting programmers was pioneered- had strong links to the military. On the other hand, several managers (in hiring and training positions) were psychologists. But no matter whether their choice was a reflection of their immediate environment or based on a more general trend, SDC’s use of psychometric tests for selecting programmers can be seen as a classic example of an organization turning to an externally available model, when the definition of a problem and its solution is uncertain (DiMaggio and Powell, 1983). This highlights the problems that industry and practitioners confronted when trying to define what skills computer workers were supposed to have.

At the same time, because of their prominent role in hiring and training an early generation of programmers, SDC played an important role in legitimizing the use of these tools in the programming field. Soon many more companies developed and adopted similar psychometric methods to evaluate the potential of candidates for programming jobs, though there was no overall agreement about either the elements to be tested or the ways of testing them. In 1962, an SDC survey of programmer selection methods in 500 organizations found that 137 (out of 250 responding) organizations used a total of 60 different tests (Perry, 1962). A few years later, a survey that elicited response from 486 organizations found that 68% of them used psychological tests in selecting programmers (Dickmann and Lockwood, 1966). They also found that, although psychometric tests were not the only selection criteria used, companies that used them tended to place less importance on other criteria like education. For example, while only 16% of employers who did not use aptitude tests accepted inexperienced candidates with a high-school degree, that percentage was 32% for employers who did use the tests. The difference was even more marked in the case of experienced programmers: only 6% of employers who did not use aptitude tests accepted

experienced candidates with a high-school degree, while the rate of acceptance in other companies was 25%.

There was considerable variety among the psychological features and mental abilities that employers hoped to detect in candidates by using these tests. Perhaps the only feature of computer work that everybody agreed about was that it required excessive mental work and therefore higher than normal intelligence. This led some organizations to believe that there is no fundamental difference between programmers and other employees and to use the tests that they were using for hiring other employees for selecting programmers (see Ensmenger and Aspray, 2002: 145). Tests that were most commonly used were those that claimed to measure skills like “verbal meaning” and “reasoning”. Candidates’ mathematical skills (in arithmetic reasoning, number series, figure analogies, etc.) were examined in some tests, but their use was controversial. Some tests focused on candidates’ ability to solve complex multi-step problems instead of mathematical skills. A wide variety of personality traits were also associated with programming skills. They ranged from creativity to detail-orientation to “emotional stability”. Employers who wanted to have a more comprehensive approach to their selection process (including SDC) used personality profiles (a combination of psychological tests, vocational interest questionnaires and personal histories) to assess the suitability of an individual (see Ensmenger, 2010a: 63-70).

The most-widely used and best-known test for selecting programmers was IBM’s Programmer Aptitude Test (PAT). Developed in 1955 (as Aptitude Test for Electronic Data Processing Machine until it was renamed in 1959), the test was a revised version of IBM’s test for punch card machine operators (McNamara, 1967). Within three years, it was reported that about 80 percent of all employers used psychometric tests in their programmer hiring process, and half of them used PAT (Lawson, 1962). In Dickmann and Lockwood study in 1966, 86% of organizations using psychological tests relied on PAT (282 out of the 486 total). The test was also used by many vocational training schools (see below) as a preliminary test to give potential trainees an indication of their (hidden) talent for programming (Abbate, 2012: 50). In the subsequent years, IBM provided alternative and updated versions of PAT, though none of them gained the popularity of PAT let alone replacing it. PAT remained the *de facto* gateway for entering programming jobs well into the 60s and even 70s (Ensmenger, 2010a: 65). In 1967 alone, seven hundred thousand individuals had sat through PAT tests (McNamara, 1967). Thus, the SDC approach to solving the selection problem had become the standard way of hiring programmers, and IBM’s test had become the main tool for doing so.

It must be noted that the scientific basis of using these methods for selecting programming candidates was questionable. As early as 1957, evidence was produced from within SDC that “in every case, the correlation between test [results] and criterion [real work performance reviews] was not significantly different from zero” (Rowan, 1957: 351).<sup>32</sup> IBM’s McNamara and Hughes (1961) claimed that there was some correlation (in the range of 0.36 and 0.44) between PAT scores and supervisor ratings but another study (by SDC) found no relationship between PAT scores and programming performance (Reinstedt, et al. 1964). Few studies found surprisingly that high scores on these psychological tests were negatively correlated with subsequent performance ratings by managers. At best, researchers found some correlation between succeeding in the test and doing well in training programs. In most cases companies did not follow up their tests with any validation study but the common belief was that these tests “were doing a good job” (Dickmann and Lockwood, 1966; Seiler, 1967). So, even though some continued to argue that results did correlate with performance (e.g. DeNelsky and McKee, 1974), the evidence for their effectiveness remains, at best, inconclusive (Ensmenger, 2010a:66-7).

Whatever the scientific validity of aptitude tests, their value for employers was soon going to be diminished. Since aptitude tests had become de facto gateways to computer jobs, many candidates concentrated on improving their test skills. Taking the same test multiple times would almost certainly lead to improved scores. Hence, “would-be programmers simply applied for positions at less-desirable firms, mastered the aptitude tests and application process, and then transferred their newfound testing skills to the companies they were truly interested in” (Ensmenger, 2010a: 72). Others tried to cheat tests by acquiring copies of them and practicing. By 1967, a textbook titled *How to Pass Computer Programmer Aptitude Tests* had appeared to help interested individuals (Abbate, 2012). Nevertheless, employers continued to use the tests, and in fact, IBM PAT was used long after IBM stopped using and distributing them in mid 1970s.<sup>33</sup>

The use of these tests has other aspects that are less commented upon: was there even an implied agreement over what the job of programmers involved? In the discussion following Dickmann and Lockwood presentation of their paper, Dickmann reflected explicitly on this point:

---

<sup>32</sup> IBM’s PAT had some supporting evidence (see O’Shields, 1965) but independent research had found no relationship between programmer performance and PAT scores in the context of business applications (Reinstedt, et al: 1964). A large number of other tests were never validated (see Seiler, 1967).

<sup>33</sup> Karten, H. (1982) ‘Pros and cons of standardized aptitude tests for programmers’, *Computerworld*, 12, Jul 1982.

“I think that the job of the programmer can vary quite appreciably from one organization to another. We noticed this in the use of an appraisal instrument we developed where some of the questions were designed to elicit information or rated a person on how many professional organizations he belongs to, how many papers did he write, this sort of thing, items which indicated that they were typically research oriented, and were rating the programmer on his professionalism, while in other cases an organization would say these questions don't mean a thing to their organization. "I want to know how good that guy can program and what he can produce for me - I am not very interested in his heading out to conferences and spending his time writing papers and that sort of thing." This would indicate the style of the supervisors who make the ratings can vary considerably in what they expect of their programmers; whether they have them go out and meet customers or whether they lock them in a closed room and tell them to program, etc. I think programming jobs can vary considerably even within one organization.” (Dickmann and Lockwood, 1966: 26)

Here, again we can observe that the drive towards commercialization of computers was accompanied by will to remove the non-business-oriented aspects of programmers' jobs (such as research and publication). This is in parallel with the initial resistance to developments of programming tools as nothing more than a useless and expensive experiment. Thus, part of the 'rationalization' of programmers' job that occurred in this period was removal of non-business related tasks from the task area of programmers. Therefore in contrast to early computer conferences that were attended by practitioners, very soon a divide emerged between those who engaged in programming research (often in universities, but also inside corporations) and those who performed the more mundane programming tasks of developing applications.

A comparison with medical occupations like physicians can shed light on the far-reaching consequences that this institutional arrangement has had for programming. On the one hand, medical research and practice (at least in the U.S.) are far more integrated and related to each other. On the other hand, even those practitioners who do not engage in research activities are trained through the same institutions that researcher work in and have to follow their developments very closely. In contrast, as I will show in the subsequent chapters, in programming (and computer work, more generally), not only researchers and practitioners are separated but also the failure of research institutions and universities to control the entry to programming work means has resulted in a fundamental gap between researchers and practitioners.

Another indication of the influence of the commercial drive behind these changes is the undefined relationship that seemed to exist between psychometric tests and the computer vendors. At least one company reported to have stopped using IBM PAT because it was no longer using IBM equipment (Seiler, 1967). More generally, it is surprising that so many computer manufacturers were actively involved in creating and disseminating these tests. IBM, Univac, Honeywell, National Cash Register, Radio Corporation of America and the Burroughs, each had their own test batteries (Abbate, 2012: 50). Given the ostensibly scientific basis of such tests, this blending of aptitude test instruments with commercial interests is puzzling, to say the least. One could argue that the most likely explanation for the success of PAT, apart from its direct simple title, should be sought in the dominance of the IBM in the market since the late 1950s.

The unsettled questions about skills, the break in the link between research and practice, and the ambiguous results of psychological tests which nevertheless emphasized mental abilities, all shaped an emerging recognition of programming as a special kind of work: programmers, it seemed, were doing something with complexity of the science but creativity of the arts (see, for instance, Knuth, 1968). Perhaps the most romantic expression of such sentiments came from Fredrick Brooks who likened the programmer to a poet who “builds his castles in the air, from the air, creating by the exertion of imagination” (Brooks, 1975: 7). This along with the uncontrolled entry of individuals to programming jobs had important consequences for the common perceptions about computer work. The success of programming projects seemed to be dependent on the idiosyncratic abilities (the “black art”) of individuals and their artisanal practices (e.g. Backus, 1980). In the next section, I challenge this perception by discussing the other aspect of computer work in SDC, i.e. the way software development was organized.

### ***5.7.2. The software factory ideal and the organization of software development***

Throughout the 1960s, programming tasks were widely believed to be too complex and difficult to be managed. Therefore, the success of development efforts seemed to rely on finding the lonely geniuses, often incapable of having social relations, who had inexplicable talent to develop software without making any error and/or use “clever tricks” to overcome the limitations of hardware that was being used (see Friedmann and Cornfield, 1989: 94-6 and 120-21). Historian Nathan Ensmenger has argued that this conception of successful programmers as geeks, which was subsequently internalized by most programmers irrespective of their capabilities and talents, was later used against their ambitions for professional autonomy as managers sought to bring their activities (and costs of programming) under control using structured programming, software engineering technics

and similar methods (see Ensmenger, 2010a: chapters 6 and 8). While it must be noted that many of these methods were not developed by managers but programmers, Ensmenger's argument is interesting because it highlights the role that cultural factors have probably played in the failure of programmers' professionalization efforts. In this context, the principles around which programming efforts were organized in SDC can be instructive.

Almost from the start, SDC managers had no doubt that programming tasks could, and had to, be controlled (see Benington, 1983). This was in sharp contrast with the common understanding of the time, and it is not clear why, unless one takes the particular blend of scientific rationality and military mentality of SDC managers as a factor. In any case, the extent of control that programming managers at SDC sought is indicative of their general style and attitude towards programmers: according to Benington (in Tropp, 1983: 387) SDC managers believed that some programming tools (like programming languages) were "too dangerous because they couldn't be well disciplined". Instead, they had developed tools that provided team managers with extensive control over the programmers:

"You could assign an individual a job, you could control the data that that individual had access to, you could control when that individual's program operated, and you could find out if that individual was playing the game wrong and punish the person."

It is unclear how forcefully this disciplinary system was imposed, even though the high attrition rate mentioned above indicates that it was creating problems, at least in the later stages of the SAGE project. Certainly the rest of industry (who were not for security reasons aware of the details of project) did not like the approach and it was called (by its detractors) the "Mongolian horde" approach to programming (Ensmenger, 2010a: 61). The cost of each line of code in SAGE was much higher than the norm of the time (Benington, 1983: 351) and it required more, rather than less, programmers. But, similar to other national-security projects, the cost of project did not cause much concern (Pugh and Aspray, 1996) and the project benefited from the pool of novice programmers who were paid to be trained.

What this description of development efforts in SDC indicates is that extensive rationalization of programming tasks was indeed possible. Surprisingly, however, it was neither associated with the use of programming languages nor lowering costs. Instead, it depended on using programming tools to enhance the control of managers over the practices of programmers. It is important to remember that in the context of SDC, managers of programming units were themselves experienced programmers. Thus, if the organizing principles of software development in SDC were adopted in the wider industry, it was possible that it could lead to establishment of a particular system of journeyman-to-master

occupational progress. This would have ensured that the same principles were used in further developments of the field, creating a stable basis for the formation of jobs and skills. At the same time, as the case of SDC shows, it was unlikely, that programming jobs were as exciting or as highly-paid as they became.

Moreover, this would have not been at the cost of innovative technologies or methods. SDC pioneered many programming and simulation tools). They also pioneered many structured programming methods that were independently developed much later in the industry. Many of these techniques were more than 25 years ahead of the industry (see Boehm's comments in Benington, 1983). While it is unclear why those programming techniques did not become popular in the industry, the reactions that it provoked from the industry suggests that its implications for programming costs were considered undesirable. Thus, again the rationality of managers avoiding those tools at the time followed their prevailing ideas about how to control computer work: cost-saving through automation had become accepted as the acceptable form of rationalization of computer work. Any method that would increase those costs seemed 'irrational'. It is important to note that ambiguities surrounding the criteria for programmers' skills and American managers' historical preference for cost reduction through automation and reduction of reliance on skilled workers contributed to these developments at this stage of the history of industry.<sup>34</sup> (see Thelen, 2004: 177).

As a non-for-profit organization funded by government and military projects, SDC was not very much concerned with costs and offered what was considered a generous but not out of norm salary. Therefore, from the perspective of novice programmers, working at SDC provided not only an exciting challenge, but also a good salary with benefits. 89% of the first cohorts of recruits were between 22 and 29 and many of them were recent graduates (Baum, 1981: 50), The nation-wide advertising campaign of SDC meant that they had to be gathered from virtually every state. For some, these benefits were too good to be true: "My mother will never believe I'm earning 350\$ a week in legitimate business" one excited recruit is reported to have said (Baum, 1981: 49).

Nearly five decades later, wife of Al Vorhaus -one of the earliest recruits who joined the programming team before SDC was incorporated- wrote in their family memoir about the excitement of that day in 1955, when her husband had returned from interview with RAND:

---

<sup>34</sup> In contrast, Japanese companies, which operated in a historically formed context of 'dampened' skilled-worker mobility and long-term career-structure, preferred the 'software factory' solution. It is notable that, although lacking evidence, participants in the 1969 NATO conference on software engineering cited Japanese practices as superior. See Cusumano (1991, especially p.41).



“Finally, in he came, incoherently trying to tell me everything at once. He had a job offer that included a generous starting salary and an all-expense move to the Boston area for training on a new-fangled thing called a “computer”. He was going to learn to “program” and become a “digital computer programmer”... “They want me for what I can do in the future, not for what I’ve done in the past! They gave me a bunch of tests and they told me I’d be a natural for the work, whatever it is”. ... I figured it was a scam...It was too good to be true”. (Vorhaus, 2000: 110)

In fact, upon completing the training course, Vorhaus passed the exam with full marks. The original contract had specified that upon completion of training he would be moved around the country to work at various sites, but thanks to his excellent record, he became the trainer for the next series of recruits. Moreover, he and his family -like other SDC programmers- were treated very well:

“We moved to a better neighborhood [in Boston] at company expense.... All moves were first class, including air fare, car shipments and animal transport if required. Nothing was too good for programmers and their families.” (Vorhaus, 2000: 114)

Some of the benefits that early SDC programmers received (like first-class tickets and a 75,000 dollar annual coffee bill which lasted until 1960) were exceptional, but benefits like paid vacation, insurance and retirement plans were increasingly common. Towards the end of 1950s, as programmers gained their artisan status, programming positions emerged from the low ranks of organizations to become highly paid and well-respected jobs. Their lavish treatment was partly a sign of recognition of these trends and partly a sign of increasing competition for programmers. SDC managers knew well that their programmers were highly sought-after, and being a non-for-profit body, they could not compete with corporations who were ready to pay more (Baum, 1981: 51). As we will see in the next chapter, the nascent computer industry of the mid 1950s drew significantly on the developments that were pioneered in the SAGE project and more specifically, on the pool of skilled programmers that SDC had created.

## **5.8. Summary**

In the first section of this chapter, I discussed the origins of programming work and placed it in its historical context. I showed that programming jobs were considered too mechanical to be appropriate for scientists and engineers. At the same time, the existing labor market provided a significant number of human computers that were considered the best candidates for performing the job because of the mathematical skills that ‘translation’ of scientific calculations to step-wise computer operations required. In the second section, I showed that when the context of programming changed, mathematical skills no longer mattered but there was no agreement over what new skills were required.

At the same time, development of programming languages, which required a significant amount of persuasion before they were accepted in the industry, created a new market for competition between vendors. While I explore the dynamics of competition in the next chapter, the purpose of discussion here was limited to show that programming languages did not simply replace mathematics as a required skill; instead, in the absence of any skill-based criteria, individuals were first admitted into the labor market and then trained to become a programmer. Computer vendors, who had huge incentives in investing in those skills, started training programmers on a short-term vocational or on-the-job training basis. This gradually changed the dynamics of skill formation, from one based in university education to one affected by the competitive strategies of corporations and the role that software workers played in those strategies.

In the third section, I explored the pioneering role of RAND and SDC as an early employer/trainer of programmers, in a non-for-profit context. The investments that were made during this project, created a significant number of programmers who were not committed to the project or their employers and therefore could quit and move to other employers at will. At the same time, recruitments at SDC, which remained focused on psychometric tests, continued to train more and more programmers from which the industry hugely benefited. I have also highlighted that, in contrast to the 'general wisdom' of granting programmers discretion and seeking cost-reduction through programming tools, SDC managers sought to develop tools that extended their control over programmers even though this approach was most costly. In the next chapter, I describe the emergence of industry (which was roughly contemporaneous with the second and third section of this chapter) and develop the issues that I have identified here in more detail.

## 6 The Rise of IBM in the computer industry: 1954-1969

### 6.1. The shaping of competition in the computer industry

In the year 1954, which marked the first conference on computing personnel and the selection of IBM for building the SAGE computers, IBM was not the leader in the electronic computer industry. The leader was Remington Rand, a manufacturer of typewriters and cardpunch operating machines that had acquired the Eckert-Mauchly Computer Corporation (EMCC), the company founded by the makers of ENIAC. In this section, I start with the story of Eckert and Mauchly's short-lived company and then explore how Remington Rand that had a five-year lead lost its position in the market to IBM.

John P. Eckert and John W. Mauchly had started up their company in 1946 (in Philadelphia, PA) with the objective of commercializing the application of computers. In order to do this, Eckert and Mauchly gathered a group of experienced programmers, including three ENIAC girls and Grace Hopper,<sup>1</sup> to develop programs that could be used to demonstrate the possibility of using computers in businesses (see Norberg, 2005: esp. 108-115 and 192-205). At the same time, Eckert and Mauchly were busy contacting potential customers. A memorandum written in January 1948, lists a total of twenty-two companies, government agencies and other institutions that Mauchly had personally contacted to persuade them to order a computer. The correspondents were generally positive, but only two orders were made (one for a custom-built computer called BINAC and the other for Eckert and Mauchly's original design UNIVAC). But EMCC soon ran into financial difficulties and, in 1951, before the first UNIVAC was delivered, Eckert and Mauchly sold their company to Remington Rand. Between 1951 and 1954, the EMCC division of Remington Rand (later known by its main product, Univac) sold twenty UNIVAC computers (Ceruzzi, 2003).

This short-lived company is particularly interesting because it presents a picture of the doubts that existed surrounding the use of psychological or similar test at this moment in the history of the industry. The various people who were at charge of hiring people in this company apparently did not use those test or at least did not use them systematically. Jean Bartik, one of the ENIAC girls who had moved to EMCC, said:

“We used to argue about what kind of a person [made the best programmer]. We had this little test that [we used to give to candidates]...Hildegard Nidecker came along, who had much experience in doing calculations for the Army. And she flunked her test, so nobody wanted to hire her. Then [Arthur Katz who later became

---

<sup>1</sup> According to Hopper's biographer, Beyer (2009: 171), in joining EMCC, Hopper had turned down offers offer from multiple military units and, most significantly, ERA (see below).

the head of Univac service bureau] said, "This is ridiculous. This just proves to me that the test is ridiculous. We know that she is going to do a good job," and in fact [we hired her] and she retired from UNIVAC [in the 1980s]."

In EMCC, it seems, there was a tension between using mathematical education or some other sort of test as a selection criteria. Betty Snyder, another ENIAC girl at EMCC whose degree was in journalism, used to ask candidates: "Did you like plane geometry?"

"I had no idea what you [should] ask for.... I figured, you know, if you liked to solve puzzles, you must be pretty good... Actually, they were looking for mathematicians but even so." (Univac Conference, 1990: 59; see also Albert Tonik's comments, p.55).

But as these comments show, the EMCC team was thinking along the same lines as the rest of (emerging) industry. In time, as mentioned before, Univac had its own aptitude test.

Remington Rand acquired another pioneering computer company in 1952. Engineering Research Associates was a company founded by a group of wartime Navy cryptographers with government support and private funding (in St. Paul, MN). They constructed their first computer for the Navy by drawing on the ongoing work at Whirlwind and other pioneering projects. According to Robert Price, ERA personnel were engineers with a highly technical engineering focus. For constructing their first computers, they relied on a close daily relationship with their clients who were onsite and involved throughout the construction process (Misa, XX: 135-6). After being authorized to market the computer, they produced five more computers until 1952. Because they had worked on a non-competitive basis and mainly through government contacts, their operation became under scrutiny and, while constructing their third computer model, they were sold to Remington Rand (Fisher et al. 1984: 10). Thus, by 1954, Remington Rand had sold a total of more than forty computers through its two divisions, gaining a significant lead against the competition.

IBM, on the other hand, was slow to initiate its computer activities. Following its involvement with the building of Harvard Mark I, IBM had some doubts about whether or not to enter the emerging computer market (Fisher, et al., 1984: 11-14). It had developed several electro-mechanical calculators for scientific purposes<sup>2</sup> but was doubtful that there will be a market for such devices in the business environment. The most significant of these calculators, from a technological standpoint was CPC. The idea for the device came from Northrop Aircraft (based in California), the same company that had ordered BINAC to EMCC and a heavy user of such machines for calculating missile trajectories. Though not digital and without capability to store programs electronically, it was the first IBM's device

---

<sup>2</sup> IBM hired Aiken's ex-student Robert Seeber in 1945 to direct the technical development of its calculators (Beyer, 2009: 85).

that could store the interim results of calculations and retrieve them as required. Soon many other engineering companies, particularly in the West Coast expressed interest in the product and deliveries began in late 1949.

**The Producer of**

# UNIVAC

**Is Expanding Its  
Field Of Operations!**

Yes . . . these are all-important facts to men who are not content to follow established patterns created by others . . . but seek, by their own creative and pioneering instinct, to lead the field through the development of even **NEWER** and **MORE ADVANCED** methods to be incorporated into UNIVAC III.

Here at Remington Rand, the first name in electronic computing, we encourage creative thinking. Where others stop, we carry on with new applications and developments. That's why only Univac, with no "extra" equipment, can check its own work, read, write and compute simultaneously. To these already proved superior accomplishments add the speed of Univac II's magnetic core storage - and you have the first electronic business data-processing system to use magnetic core storage successfully.

**To Qualified Men  
THE REMINGTON RAND  
UNIVAC DIVISION  
Offers  
Permanent, Well Paying  
ENGINEERING  
POSITIONS**

## ENGINEERS PHYSICISTS

**The  
FIRST NAME**

★ **In Complete Electronic Computing  
Presents**

### OPPORTUNITIES UNLIMITED

**To Experienced Men . . or Those  
Recent Graduates  
Who Can Qualify As**

- ▶ **Electrical Engineers** . . . Circuit Designers to develop and apply new circuit techniques. Degree in E.E. or equivalent experience required.
- ▶ **Mechanical Engineers** . . . to design and develop electro mechanical devices; electronic packaging design and development.
- ▶ **Physicists** . . . for fundamental work in the field of ferromagnetism and development in Solid State components; design of optical systems; development of phosphors and ferromagnetics.
- ▶ **Logical Designers** . . . for system design in effecting logical functions of electronic data processing equipment.

The above openings lie in the field of electronic or electro-mechanical data processing equipment, research and development and specifically in such areas as Magnetic Tape Mechanisms . . . Paper Handling Devices . . . Magnetic Drum or Tape Applications . . . High Speed Mechanisms . . . Transistors . . . Electronic Packaging Design . . . Servo-Mechanisms and many other areas of interest.

<ul style="list-style-type: none"> <li>▶ <b>Mathematicians and Programmers</b> . . . with experience in electronic digital computer programming or extensive math background.</li> </ul>	<ul style="list-style-type: none"> <li>▶ <b>Metallurgist or Physicist</b> . . . for work in Magnetic Alloys; strong background in Magnetic Alloys required.</li> </ul>
--	--

- ▶ **Product Analysts** . . . experienced in customer application of business machines or related equipment; applications analysis.

In addition to the challenge of being associated with the leading men in this field, our qualified employees benefit by a liberal cooperative educational aid program . . . health and hospitalization benefits for BOTH you and your family . . . progressive retirement plan . . . moving and travel expenses.

WRITE OR PHONE  
**D. A. BOWDOIN**  
Placement Manager  
BALDWIN 3-7300



## Remington Rand

**UNIVAC DIVISION**  
SPERRY RAND CORPORATION

2300 W. Allegheny Ave. Philadelphia, Pa.

Figure 6-1: Univac's ad. Programmers must have experience or "extensive math background".

Remington Rand offers "Permanent, well paying engineering positions" 1955

In 1952, IBM finally announced its first digital computer IBM 701 dubbed as an “Electronic Data Processing Machine”. Thus, it was only a few months after the production of 701 that the MIT team selected IBM for building computers for the SAGE project. By building on what they had learned in SAGE project, IBM tried to catch up with Remington Rand and delivered three consecutive models including a particularly successful model 650 (see Watson and Petre, 1990: 259). Also, IBM quickly became aware of the potential of rising software costs to negatively affect customers’ willingness to invest in computers. At the initiative of one of IBM’s sales managers, a computer users’ group called the Digital Computer Association was formed in 1952 in order to facilitate communication amongst IBM’s customers and between IBM and its customers. Through this association, member companies shared their experiences as well as pieces of software. Soon this collaborative effort resulted in development of a new programming system called PACT that was better than IBM’s own package of programming tools (Campbell-Kelly, 2003: 32).

IBM assigned the first 701 to its Technical Computing Bureau (part of the Applied Sciences department) in December 1952 and started training 6 months before deliveries were made to the customers. Beyond basic training, IBM encouraged its trainees to start working with customers and develop programs before their computers were delivered. The Technical Computing Bureau became the center of customer support over the next years. There were about 30 programmers assigned to the bureau, all with “at least a bachelor’s degree in mathematics”, half with master and a few with doctoral degrees (Pugh, 1995: 188) Within the first year, near one hundred application programs and forty system programs were developed by the bureau staff.

Meanwhile at Remington Rand, the overlap between the product market of EMCC and ERA had created some degree of competition and friction between them. Still, this remained a minor issue because each division had its own preferred market: Univac was mainly active in business (or data-processing) applications while ERA preferred scientific and engineering applications (see Ceruzzi, 2003: 37; Misa, 2013: 66-67). Along the same lines, an important distinction between the two was that key members of ERA were hardware-focused engineers but Univac division employed several experienced programmers. Whereas ERA division focused on customers who were capable of developing their own programs, programmers played a prominent role in the Univac division. According to Beyer (2009, 253):

“The customer service bureau [of Univac]...helped potential clients determine their computing needs.... Once defined, the problem was sent to Harper’s team in Philadelphia, and two or more programmers worked on it for a month or two. The

finished program was then run for the potential customer [on one of the available computer machines]. If the demonstration was a success, the service bureau followed up to sign a contract.”

This strategy put software work at the forefront of development activities and made it a crucial factor in the success (or failure) of division. As mentioned in the previous chapter, in 1953, Remington Rand had set up a programmer-training center in its sales office in New York (see previous section and Beyer, 2009: 252-3). Also, at about the same time as SHARE was set up, Remington Rand created its own user group Univac User Association and later Univac’s Scientific Exchange (Univac Conference, 1990: 53; Campbell-Kelly, 2003: 33). But still many more programmers were needed for providing customer support and, even worse, some programmers were lured away by clients who wanted to expand their own programming effort. By 1954, Hopper’s programming team was suffering from a high attrition rate and Mauchly asked Remington Rand managers for more budget to be able to offer programmers better salaries. Mauchly warned that “We are not only losing the battle of hardware, but the battle of applications research and education” Fisher, et al. 1984: 41). But Remington Rand management was unwilling to spend much in research and development and certainly did not want to pay a higher salary to programmers (Beyer, 2009: 255, Fisher, et al. 1984: 22). Mauchly’s request was turned down.

Remington Rand managers’ apparent lack of appreciation of the problems that the two divisions were facing and their intrusive management style created problems for both divisions but still, at least, both divisions were able to continue to work independently (Norberg, 2005: Ch. 5; Beyer, 2009: 208-212, Misa, 2013: 64-7). Problems were multiplied, however, when in 1955 Remington Rand was bought by Sperry and became part of newly formed Sperry Rand. The new Sperry Rand management decided that the two divisions had to be merged and become one division under the title of Univac. Confusion and infighting ensued and soon key employees (especially from ERA) left Sperry Rand (Misa 2013, 73; Fisher et al. 1984: 45-6). Univac division continued to operate but with limited management support and under financial pressure. Thus, by 1957, Sperry Rand had lost its initial advantage and had become one of the several smaller companies who were competing against the new industry leader, IBM. According to Remington Rand historian, Arthur Norberg, it wasn’t until 1960 that Univac division was stabilized (Norberg, 2005: 223).

During the same period that Remington Rand was struggling with organizational and technical problems, IBM expanded its production lines and activities in the computer field. The success of Digital Computer Association led to establishment of SHARE, which



for graduates (Bachelors, Masters, or Ph. D.) in  
**MATHEMATICS or PHYSICAL SCIENCES**  
 in the area of Applications for the most advanced Electronic Digital Computers.

EXCELLENT POSITIONS AVAILABLE for both experienced and inexperienced in

- COMPUTER PROGRAMMING
- NUMERICAL ANALYSIS
- TECHNICAL WRITING
- DATA PROCESSING

**IF** your answer to these questions is yes, call PL 3-1900, Ext. 8329, or write to:  
 Mr. E. S. Kopley, Data Processing Center,  
**THE SERVICE BUREAU CORPORATION, 590 Madison Avenue, NY 22, NY**

- Would you like to get into one of the most challenging, productive and satisfying professions?
- Would you like to work on some of the most interesting problems of our time?
- Would you like to work in an academic atmosphere for an industrial organization?
- Would you like to join one of the fastest growing computer organizations in the nation?
- Would you like to go through one of the best training courses in the country for computer programming?
- Are you interested in rapid, sound, personal, professional development?
- Are you interested in planning a long-range career?

Figure 6-2: IBM's Service Bureau Corporation's ad, 1957.

officially became active in 1955 with 14 member organizations. Within a year, it had a library of nearly 300 programs, 62 members (some from outside U.S) and 88 subscribers who received updates about programming literature. In 1956, charged with monopolistic behavior by the government, IBM agreed to a consent decree that, among other things, required it to operate its service bureau division as a separate corporation (called the Service Bureau Corporation or SBC). Even though this created some problems in terms of reshaping the organization of IBM, it does not seem to have affected the performance, or indeed the competitive strategies of IBM (see Elliott, 1964).<sup>3</sup>

<sup>3</sup> Elliot, R (1964) 'Program for growth? Computer service firms may have found the solution to their own problems' Barron's National Business and Financial Weekly, 23 Mar 1964;



In 1957, William Norris, head of the Univac Sperry Rand left Sperry Rand with two other colleagues to found Control Data Corporation (CDC). They were all former members of Engineering Research Associates and initially started CDC as a “hardware-only” manufacturer. Nevertheless, CDC delivered its first computer in 1960. Their strategy was to continue in the tradition of ERA and focus on engineering or scientific applications that required a high-performance computer but little programming support, thus carving out a niche for their computers. But soon they realized that in order to sustain growth, they needed to widen their customer-base and provide more services.

Soon, CDC expanded its software activity by establishing a software group<sup>4</sup> in Palo Alto in 1960. Its director, Richard Zemlin, was a mathematician and former ERA and Univac employee. Under Zemlin’s direction, CDC customers formed a user group called ‘Co-op’ (similar to IBM’s SHARE) for sharing the results and experiences of their programming activities. As part of Co-op, customers created, among other programs, an operating system (with a set of compilers and an assembler) called Co-op Monitor. Also in 1960, CDC opened its first data center in Minneapolis (MN). The idea behind data services was to simply provide computer time to customers who could not afford to buy a computer or needed additional computational power at specific points in time. Over time CDC learned that, customers using data centers needed programming support and therefore data services provided programming support in addition to computing time. This was a successful strategy and by 1965, CDC had data centers in seven U.S. cities including Palo Alto and Los Angeles (CA), Cincinnati (OH), Chicago (IL), Huston (TX) and Boston (MA) and Washington. In order to increase the computing power available to any customer, these computer centers were connected to each other (forming “Cybernet”).

Apart from data centers, CDC had a growing number of customers. Every new sale of computer required additional software work. As it was common in this period, a typical term of after-sale services was that CDC would provide free programming support, often in the form of one programmer or more on the site of the customer<sup>5</sup> for six months or more. Programmers were assigned to the customer site along with a sales representative and, together they would provide a mixture of marketing, training, programming and support

---

<sup>4</sup> This software group remained independent of the Application Services (below) and also the data center that was located in Palo Alto.

<sup>5</sup> For instance, one early customer was the Kitt Peak National Observatory (in the mountain ranges of Arizona Sonoran desert. They could not afford to have a ‘resident programmer’ and CDC had to fly in and out its programmers to provide services (Misa, 2012 :46).

services. The expansion of CDC customer base (either in new computer installations or expansion of data services) required hiring more and more programmers. In one year (1960-61) the number of programmers in Palo Alto center grew from less than 10 to 40-50 people. Sometimes programmers were poached from other companies. Many early programmers, including the first manager of the pioneering data center in Minneapolis, Richard C. Gunderson, were attracted from Univac. According to Zemlin, California aerospace industry at the time provided “a hotbed of programming talent” and CDC expanded by tapping into this pool.

In 1962, the widely dispersed support services of CDC were consolidated under an ‘Application Services’ division. The head of the division was Robert M. Price, who was hired initially as a programmer by Zemlin and went on to become the successor to William Norris (Misa, 2012: 39). Price was responsible for coordinating programming support, customer training and documentation activities (Misa, 2012: 61). One of the difficulties that this new division was set up to overcome was that services provided for each customer were contract-specific. There was no company-wide set of tools to be used and “every single customer” required “a unique set of software” (Schumacher, 1982; quoted in Misa, 2013: 119). The application services group made sure that when necessary, people from all software groups (in different data centers) were made available to satisfy the terms of contracts. As a result, many programmers used to fly from site to site.

Several other companies entered the computer industry in late 1950s but their market share and influence in the industry remained limited. Honeywell, for instance, entered the computer industry by purchasing the (relatively) inactive computer division of Raytheon that had manufactured only one computer.<sup>6</sup> As early as December 1955 Honeywell had established its Datamation Division in Wellesley (MA) to provide training and programming services but until 1960, Honeywell sold only five more computers (Misa, 2013: 145). Another company that entered the market by purchasing an existing computer firm was Burroughs. In contrast, National Cash Register (NCR), General Electric (GE), and Radio Corporation of America (RCA) all had over time developed internal capabilities to build and market computers (Flamm, 1988). Nevertheless, none of these companies affected the market significantly till the early 1960s.

---

<sup>6</sup> Raytheon had benefited from hiring a number of Aiken’s students, including Robert Campbell and Richard Bloch, in 1947 (Norberg, 2005:95). Campbell moved to Burroughs in 1949 (Beyer, 2009: 163).

Thus, in the early 1960s, the main players in the computer industry were IBM, Univac and CDC (Fisher et al., 1984: 65). In the same period that these companies and their smaller rivals competed in the mainframe market, a number of small companies had emerged to provide a variety of computer-related services, including: programming services, data processing services, and facilities management.<sup>7</sup> These start ups were typically founded by enterprising individuals who typically did not even own a computer. Nevertheless, many of them had learned systems analysis and programming in one of the vendor companies or user organizations, or at least had access to a pool of such individuals. The more successful of these start-up companies had their own computers and employed tens of individuals. For example, the Computer Usage Company (CUC) -generally regarded as the first programming services company- was created by two former IBM employees: John Sheldon, former director of IBM's Technical Computing Bureau and Elmer Kubie, a programmer in that bureau. They started their company in New York by renting a computer from IBM and employing four female programmers. By 1962, they had 240 employees and branches in Washington and Los Angeles.

Type of Service	Description of services	Access to computers
Programming	Developed system or application programs according to a contract for a specific customer	Usually through customer but some owned
Data Processing	Processed customers' data on a regular basis	Owned
Facilities Management	Managed the operation and maintenance of customers' computer systems	Customers

Table 6-1: Different sectors of the software and services industry.

In the data-processing category, Automatic Data Processing (ADP) was a company providing processing services, which had started in 1949 but did not use computers in its operation until 1961. Before using computers, they performed calculations (for payrolls, etc.) manually or using cardpunch machines. But when they brought in a computer to perform those tasks they realized that they needed programmers to set up computers for each customer. Similarly, most of the data processing companies did have in-house programmers and unless the customer already had programming skills (like a small engineering firm) they depended on in-house programmers for smooth operation of their processing tasks. ADP soon found rivals, namely Computing and Software Inc. and University Computing Company (the latter provided part of its services over phone connections, known as "teleprocessing").

<sup>7</sup> A number of defense contractors and aerospace companies started to offer programming services, but they are not typically considered part of the software industry (see Campbell-Kelly, 1995; cf. Campbell-Kelly, 2003: 45-6).

Both data processing companies and facilities management companies were competing against the service bureaus of large manufacturing companies. The main difference between companies providing processing services and those providing facilities management services was the ownership of computers and location of offices. Whereas data processing had their own computers and offices, facilities management companies operated their customers' computers in customers' offices. Although data processing companies could provide facilities management services, this was not their priority. Few companies defined facilities management as their main field of activity. These companies usually hired skilled programmers and other workers to operate and maintain customers' computers on their behalf. By far the most successful of these companies was Electronic Data Services (EDS), which was established in 1962 by former IBM salesman Henry Roos Perot.

Some of the early start ups built on establishing a relationship with one of the vendors. For example, Advanced Computer Techniques (ACT) was founded in New York by Charles P. Lecht on the basis of developing a compiler for Univac. Lecht was a former IBM programmer who had participated in SAGE projects and who had gained some reputation by writing textbooks on programming. He started as a one-man company but following the award of contract from Univac started hiring programmers. The company grew rapidly afterwards and became involved in several compiler projects (Schachter, 2004). Similarly, Applied Data Research (ADR) was co-founded by former IBM and Sperry Rand employees in 1959 and provided programming tools for manufacturing companies including a compiler for RCA.

Perhaps the most successful company that was founded in this period was Computer Sciences Corporation (CSC). CSC's first project was to develop a compiler for Honeywell. The co-founders used their reputation (two senior member representatives in SHARE and a data processing manager from C-E-I-R) to convince Honeywell to sign a contract with them. The project that was considerably more sophisticated than the normal compilers of the time was not completed according to the plan. Nevertheless, CSC gained reputation by participating in a high-profile project, while for Honeywell, the partnership was a source of prestige because of its state-of-the-art objectives. By 1966, it had 2000 employees and competed for software contracts with major computer companies.

Throughout the late 1950s and early 1960s, most of the software companies that were founded to provide programming services gradually diversified and started providing other types of services. The primary motive for diversification came from the need to stabilize

level of operations and staffing in the company. As programming contractors, the success of these companies relied almost entirely on the abilities of their programmers, so much so that they were sometimes called “think factories” (e.g. Elliott, 1965)<sup>8</sup>. But because of the uncertainties associated with the bidding and contracting process, they were always facing the possibility of having more or less software workers than they required. Moreover, the small-scale and specialized characteristics of programming services that were referred to these companies meant that they experienced significant fluctuations in their activities. In the words of CUC founder, Elmer Kubie, they were “primarily a ‘job shop’ for programming and analysis” (quoted in Campbell-Kelly, 2003: 73). Diversifying into other services allowed them to become more stable and establish their position in the market for various services.

One of the most common strategies for diversification was acquisition of smaller companies. For instance, C-E-I-R (standing for Council for Economic and Industrial Research) was established in 1952 as a non-profit research organization to provide consulting services in modeling and operational research. When the founders of C-E-I-R became aware of the capabilities of computers, they rented one to use it in solving their problems. Soon after, they hired an experienced programmer to manage the computer operations. In 1954, C-E-I-R became a for-profit company providing programming and processing services and started to hire and train people. Soon it grew enough to acquire a number of smaller companies and a training institution, named Automation Institute, which provided computer-related courses throughout California and in Chicago. University Computing Company (UCC) adopted a similar strategy a decade later. Based in the campus of Southern Methodist University in Dallas, TX, UCC started with the idea of simply selling computer time but soon provided data processing and software services. In order to do so, it hired programmers by training university students. UCC’s subsequent strategy was to find and acquire small companies that had developed scientific or engineering applications for specific companies (like in oil or aviation industry) (Wyly, 2002). Soon it provided services in programming, processing and even leasing of computers.

Although there is no accurate statistics about the number of early ‘software houses’, an estimated 50 were active in the industry by 1965 (Campbell-Kelly, 2003: 63-4). The largest of these companies, CSC and CUC, were closely followed by C-E-I-R, Planning Research

---

<sup>8</sup> Elliott, R (1965) Flourishing think factories: Computer service companies are coming up with the right answers. *Barron's National Business and Financial*, 20 Sep 1965;

Corporation, Computer Applications Inc. and Informatics Inc.<sup>9</sup> Unlike data processing and facilities management services which faced constant competition from vendors' service bureaus, these companies were mainly competing against each other and only occasionally faced direct competition from vendors (for large government contracts, for example). Gradually, software houses started to provide software packages as an alternative to software packages that vendor companies freely provided with their computers. Software companies usually claimed that their software packages performed better (in some respect) and therefore even though they were not free, customers would benefit from the better performance of the software. Still, the scope of these software packages were limited and were usually more system programs rather than applications.

Two other developments in the same period led to a slight differentiation in the industry: 1) the rise of minicomputer market that followed the pioneering products of Digital Equipment Corporation (DEC) and 2) the rise of leasing companies. Targeting a small community of technically savvy users, DEC's strategy was to provide minimal personal support. Instead, it actively provided detailed documentation and instructions that would enable its potential customers to assemble and use their computers on their own. It also provided a few programming tools which would leave the task of developing applications to users. Throughout the 1960s, the market for minicomputers and related services remained small, as did DEC's influence in the overall computer industry, even though it became the third largest computer manufacturer (after IBM and Univac, see Campbell-Kelly, et al. 2013: 219). Soon it was joined by a number of other companies which each gained a small share of the minicomputer market (e.g. HP, Data General and Prime). The leasing companies, on the other hand, thrived by offering IBM computers at lower prices and terms more favorable than those offered by IBM. These companies used some innovative strategies (like providing solutions by combining hardware/software solutions from different vendors in the same contract) and grew throughout the decade, but their main effect was increasing pressure on IBM.

In the early 1960, compatibility (between successive models) came to be as important as technical improvement (Brock, 1975: 49-51). The rapid advances in computer technology and diversity of vendors meant that there were no (or very few) commonalities between computers of different models. So, for instance, if a company was using a computer for payroll calculations, with every computer upgrade it had to re-program their payroll

---

<sup>9</sup> Systems Development Corporation, which was almost as large as CSC, remained a non-fo-profit corporation until 1969. During its transition period (1964-69) it lost almost a fourth of its employees (reaching 3000 from 4300).

software. From customers' perspective, this lack of compatibility meant reprogramming and possible need for data conversion whenever computer was updated. Although this was not necessarily very costly (if the vendor provided free programming), still it was time consuming and could create disruptions in the organizational processes. At the same time, it meant that every customer could, in principle, change its vendor whenever it decided to upgrade.

This uncertainty presented both an opportunity and a threat for vendors (in terms of expanding or losing their customer base). On the one hand, salesmen of competing vendors would emphasize the novelty and 'state-of-the-art' features of their new models against the rivals' and, by some persistence, they could be successful in persuading the customer to switch vendors. On the other hand, to protect their customer base, vendors had to facilitate the process of upgrading to the new model. Vendors had tried to achieve compatibility between their models but this was not always possible. Therefore, software programs that could convert data or programs from one computer model to another became increasingly important.

In 1963, Honeywell made an inroad into IBM's most successful computer (1401) by developing software called, quite suggestively, "Liberator". This program offered IBM 1401 customers the opportunity to convert ('translate') their programs directly to programs for the new Honeywell 200, thus (almost) removing the need for re-programming. This feature, combined by Honeywell's lower price and better performance, attracted many IBM customers (300 hundred IBM customers switched within a year). IBM responded by providing its programmers with "Captivator", a card box of subroutines that they could use to hamper Liberator conversions (Misa, 2013: 145). But Honeywell 200 had already made an impact: not only it had improved the market position of Honeywell; it made IBM accelerate its efforts in developing a fully compatible line of computers (see Pugh, 1995: 273; Fisher et al., 1984: 134, 236; Campbell-Kelly, et al., 2013: 128).

Since 1961, IBM had become aware of the value of compatibility between its own computer models, even though it had underestimated the probability that competitors would use cross-compatibility effectively against it. It had started working quietly on a series of computers that were compatible by design. These computers, collectively known as System/360, were announced in April 1964. The range of products offered and the compatibility ambitions that were expressed in the announcement shocked the industry. Even though System/360 was a technical advance over most of the existing computers (in terms of performance), its built-in compatibility was more impressive in the eyes of many in the industry. It seemed that, in one

move, IBM wanted to both make most of the existing computers obsolete and create a new way of competing in the market.

System/360 was enormously successful and created a new challenge for IBM's competitors. Most found themselves in a reactionary position, having to decide whether to adopt IBM's approach or find another way of competing. Honeywell, which was already working towards compatible designs, continued the expansion of its "product family" by announcing new products between from mid-1964 to early 1965. RCA announced its own series, designed to be compatible with IBM System/360, in December 1964. Burroughs and NCR announced their series in, respectively, 1966 and 1968. Sperry Rand seemed to be initially unimpressed, but they too eventually started producing compatible lines of computers. GE was perhaps the only company that did not make a concentrated effort towards compatibility, but nevertheless focused its efforts on developing time-sharing system, an area in which System/360 had significant limitations. CDC's delayed response was to change its target market, focusing on superfast computers and service bureaus rather than competing in the mainstream computer market. In 1967-8, CDC acquired C-E-I-R to expand its services operation and gain access to the skills of its hundreds of employees.

The success of System/360 in warding off IBM's traditional competitors, gave rise to new forms of competition: a large number of hardware manufacturers started producing System/360 compatible components at a lower price. Although IBM managers had anticipated such a possibility, they had underestimated its upsurge following the arrival of System/360. Moreover, towards the end of the 1960s, plug-compatible companies received an unexpected boost from the Federal government. When a study by the General Accounting Office found that a number of private companies had achieved significant savings by using plug-compatible equipment, several federal agencies issued policies that required their offices to replace original IBM components with those produced by plug-compatible manufacturers. Though the extent of these changeovers is not clear, Fisher, et al. (1984: 288) claim that they resulted in "government-wide peripheral equipment replacement programs".

Nevertheless, near the end of the 1960s IBM's dominant position in the computer market was hard to challenge. It had more than 70% of the mainframe market and was a highly profitable corporation. Other companies had between 2 to 5% each and were hardly making any profit. Beyond IBM and CDC, Honeywell, NCR and GE had also extensive service bureaus (Fisher, et al. 1984: 316). The direct effects of IBM's System/360 in the services market seem to be ignorable. Customers still needed programming or processing services and most applications were custom-made meaning that they could not be used elsewhere.



However, two successful applications (a flowcharting software named Autoflow and a file-management software called Mark III) were developed (both independent of System/360) in the mid 1960s that helped change this perspective. This gave rise to introduction of several more application packages dubbed as “software products”. In the late 1960s the market for programming services and products suddenly boomed and a flood of newly established companies entered the competition, heralding the arrival of the software industry. By 1968, an estimated 3000 companies were active in programming services (Campbell-Kelly, 2003: 64). Majority of these software companies had a short life and many of them left no trace. Some of them however, endured and even grew to become powerful actors in the industry in the next decade.

## **6.2. Analysis of the competition**

When IBM entered the computer market, it neither was the prime mover nor had distinct technical knowledge. And yet in less than 5 years it gained a dominant position and was able to hold on to for more than a decade (and beyond). Few have argued that IBM’s products were inherently superior to others; there were no agreed criteria for evaluation of those computers. Many historians and industry analysts have tried to ‘explain’ the success of IBM by drawing attention to various aspects of its operation. These include: its facilitated access to knowledge as the main contractor of the SAGE project, its long-standing relationship with business customers, its sales strategy and marketing activities, and its extensive training program. Although there is no place here to discuss these issues in detail, I review each of these factors in turn with the objective of highlighting the role of skilled workers in each case.

Nearly everybody agrees that IBM benefited from having access to the Whirlwind project and SAGE design. What has proved more controversial is the extent to which IBM benefited. This issue was at times at the center of proceedings of the 1969 antitrust case by US Department of Justice against IBM.<sup>10</sup> The government team insisted that IBM had benefited significantly from its privileged access, while IBM’s lawyers insisted that IBM had taken risk by participating in the SAGE project (see Fisher, et al. 1984: 26-30). In retrospective, most historians agree that IBM’s access to Whirlwind project did indeed boost its power in the computer market both in terms of revenues and in terms of knowledge and

---

<sup>10</sup> Winkler, C. (1979) ‘IBMer says Military CPU offered innovations’, *Computerworld*, 15 Jan.

skills (e.g. Flamm, 1988: 87-90; Ceruzzi, 2003: 53; Beyer, 2009: 258-9;). Perhaps the best supporting evidence can be seen in Watson Jr.'s own words:

“[IBM’s director of Electronic Data Processing Machines] took what we’d learned working for the Air Force on SAGE and used it to skip a grade, so to speak, in computer development...In a little over a year we started delivering those redesigned computers. They made the UNIVAC obsolete and we soon left Remington Rand in the dust.”

Whether or not one agrees with technological superiority of IBM’s redesigned computers (model 702), Watson’s ‘skipping a grade’ remark shows that IBM felt they had achieved knowledge and skills without much effort. There can be no question that IBM engineers who participated in the SAGE project including Ken Olsen (founder of DEC), learned enormously from the experience and incorporated it in their future work.

IBM’s longstanding relationship with customers could contribute to its success in different ways: 1) identifying its potential customers easily 2) drawing on the mutual trust built through the years which facilitates persuasion of customers 3) knowing customer operations and what they need both in terms of technology and additional services. This may have played in favour of IBM against such companies as EMCC (before it joined Remington Rand), Honeywell, GE and RCA who did not have many contacts with user companies before entering the computer industry. On the other hand, IBM shared this advantage with NCR, Burroughs and, above all, Remington Rand whose rivalry with IBM went back to the start of 20th Century. Thus the advantages produced by these relationships can only be assessed in the light of the sales and marketing function of these computer companies.

One factor in IBM’s sales strategy that probably contributed to the ascendance of IBM’s image as a reliable provider of after-sale services was the fact that it signed lease contracts with most of its customers. In doing so, it drew on a well-established IBM tradition with which IBM customers were surely familiar. Leasing meant that rather than paying a large sum of money upfront, customers paid a fixed amount of monthly rent. IBM remained the owner of the facilities and if unhappy customers simply had to notify IBM (within the terms of contract) to remove the computer. In this arrangement, since installation and programming of computer took place in the first months of lease, any problem in this period was more likely to be tolerated because customers had not paid a substantial amount of money. In fact, some customers did not pay rentals until the programs were running. IBM,

on the other hand, relied on its robust financial status to avoid running out of money.<sup>11</sup> None of the other companies (with the possible exception of NCR) had this combination of established customer-base and financial power. Univac in particular, as the initial leader of industry was under financial pressure and could not adopt IBM's sale strategy. In this context, cases like GE Appliances' UNIVAC installation -which was fully paid for in the amount of one million dollars- damaged Univac's reputation. It must be noted that, in the long run, leasing generated much more revenue for IBM. Subsequently renting became a common practice in the industry.

At the same time, there is ample evidence that suggests, compared to IBM, organization of sales and services (including training and programming) at Remington/Sperry Rand were ineffective and disintegrated. For example, Remington Rand's cardpunch salesmen were not only separate from and unable to market UNIVACs; they would lose commissions if a Univac replaced a Remington Rand machine (Fishman, 1981: 46-7; Fisher, et al. 1984, 40-41; see also Univac Conference, 1990: 8-12; Norberg, 2005: 248; Beyer, 2009: 216-20). IBM, on the other hand, had much more expertise and a well-integrated approach. Writing in 1955, Mauchly warned "We are not only losing the battle of hardware, but the battle of applications research and education":

"[We] have done almost nothing to provide a large staff of branch-based people who are familiar with UNIVAC applications and able to advise potential customers, or help actual customers...[while] IBM has made an effort to provide not one but several [of such] representatives in each of their major branches." (Quoted in Fisher et al., 1984: 41-2)

IBM's sales representatives not only continuously looked for new customers who could be persuaded to use computers, they did so by using their long term relationships with companies to gather information about how they did their various business operations and coming up with ideas about how computers could be used. In doing so they effectively acted as proto systems-analysts. While the consistency of IBM's sales force shall not be overemphasized (see Wyly, 2008: 49; Univac Conference, 1990: 90-2), it seems that a key factor in the success of IBM sales and marketing departments is that IBM's salesmen were comprehensively trained about the computers. At the surface, this was a common strategy amongst manufacturers.<sup>12</sup> However, a closer look shows that IBM played a much more

---

<sup>11</sup> Because IBM was paid upfront for the SAGE project, 80% of its revenue from stored-program computers between 1952 and 1955 had come from the SAGE project (Pugh, 1995: 219).

<sup>12</sup> Even at a programming company like CEIR, where salesmen were not technically trained, programmers were involved in the sales activities including in writing up the contract (Robinson, 1988 :24-5).

active role not only in the training of its own employees (including salesmen) but also training of a workforce that would become the main vehicle of expansion of the computer industry.

## JOIN THE RCA BREAKTHROUGH IN ELECTRONIC DATA PROCESSING

RCA . . . world leader in electronics . . . is currently expanding its electronic data processing operations as a result of one of the most significant breakthroughs in modern electronics—the all-transistor RCA 501 system. Already the RCA 501 is being talked about as the world's most efficient electronic data processing system; its sales curve is slanting sharply upwards.

If you have experience in EDP sales or technical services, and are ready to step up to more challenging and rewarding assignments, investigate today the many new


career openings at RCA. Current positions, dealing with medium and large scale systems, include the following:


**EDP SALES REPRESENTATIVES**—Background should include a thorough systems knowledge and at least one year of field experience with either government or commercial clients.

**EDP PROGRAMMERS AND METHODS ANALYSTS**—Local openings for qualified men to work closely with both customer and sales personnel in the development of specific applications, related procedures, and programs.

*For a strictly confidential interview with RCA management, please call collect or send a resume of your background and personal qualifications to:*

Mr. F. T. Flanagan, WOODLAWN 6-3300  
Electronic Data Processing Division  
RCA, Dept. HQ-120B  
Bldg. 10-1  
Camden 2, N. J.



 **RADIO CORPORATION of AMERICA**  
ELECTRONIC DATA PROCESSING DIVISION

## JOIN THE RCA BREAKTHROUGH IN ELECTRONIC DATA PROCESSING

RCA . . . world leader in electronics . . . is currently expanding its electronic data processing operations as a result of one of the most significant breakthroughs in modern electronics—the all-transistor RCA 501 system. Already the RCA 501 is being talked about as the world's most efficient electronic data processing system; its sales curve is slanting sharply upwards.

If you have experience in EDP sales or technical services, and are ready to step up to more challenging and rewarding assignments, investigate today the many new


career openings at RCA. Current positions, dealing with medium and large scale systems, include the following:

**EDP SALES REPRESENTATIVES**—Background should include a thorough systems knowledge and at least one year of field experience with either government or commercial clients.

**EDP PROGRAMMERS AND METHODS ANALYSTS**—Local openings for qualified men to work closely with both customer and sales personnel in the development of specific applications, related procedures, and programs.

*For a strictly confidential interview with RCA management, please call collect or send a resume of your background and personal qualifications to:*

Mr. F. T. Flanagan, WOODLAWN 6-3300  
Electronic Data Processing Division  
RCA, Dept. HQ-120B  
Bldg. 10-1  
Camden 2, N. J.




 **RADIO CORPORATION of AMERICA**  
ELECTRONIC DATA PROCESSING DIVISION

Figure 6-3: RCA's double ad for programmers, methods analysts and sales representative. Sales representatives should have "thorough systems knowledge", 1959.

### **6.3. The importance of vendor-provided training**

As early as 1950, EMCC application group was providing training in programming to the newly hired programmers as well as engineers, sales personnel and customers (Norberg, 2005: 113). When the Univac (Remington Rand) sales office was set up in New York in 1953 and training activities began, Univac's newly recruited marketing and sales personnel received the same training that programmers received (Univac conference, 1990: 11). But only EMCC salesmen and not all the Remington Rand sales personnel received this training. There were also conflicts about who should provide the training, at least in the first year (Beyer, 2009: 216-8).

IBM on the other hand, had provided its first internal training course on computing techniques in late 1940s. The list of IBM staffs that were trained in this period included James Backus (future head of FORTRAN developer team) and Edgar Codd (future developer of relational database model). These in turn trained the next generation of programmers who were trained to accompany CPC installations. In 1949, IBM created a new department called "Applied Science Department" to promote "technical computation" (Pugh, 1995: 157). Cuthbert Hurd, the head of Applied Science department, gathered "a group of highly qualified people, many of whom had Ph.Ds. in mathematics, engineering or science". Subsequently,

"[They] were assigned to branch offices where they assisted salesmen in the sale and installation of CPCs and subsequent computers for solving scientific and engineering problems. The unique training and experience of Applied Science representatives enabled them to help customers in understanding and solving problems. They also advised the company's product development groups of new requirements." (Pugh, 1995: 158)

Thus, the training of salesmen and programmers in IBM was intertwined from the start. Later, when programmers were needed for providing support to customers in their installation of 701 computers, IBM's Technical Computing Bureau (part of the Applied Sciences department) was assigned to the task building on the considerable experience that had been gathered within the department. Campbell-Kelly (2003: 30) argues that "IBM's awareness of the need to provide programming support was in large part a legacy of its punch-card accounting machine tradition." While that is true, it can be argued that the legacy of its earlier experiences with users of scientific calculators which led to a high degree of integration between programming and training services with the sales and marketing activities played an equally important role. By 1958, when computers had become the main

## **A CHANCE TO FOCUS ALL YOUR TALENTS**

Perhaps you're the sort of person whose interests are so varied that you haven't as yet found a career that makes full use of all your talents.

Perhaps you have the type of mind that analyzes . . . discerns order in seeming chaos . . . gets to the heart of the problem . . . instinctively organizes things in a systematic fashion.

If so, you may be one of the men—or women—we're looking for to train as a Computer Programmer . . . though you may not know right now what computer programming is.

A Computer Programmer works with scientific, business, and government problems. After analyzing these problems, he translates them into a language intelligible to the electronic computer. The result: The computer solves the problems faster and more accurately than was ever before possible.

**AN IBM PROGRAMMER'S** assignments vary. He may program a computer weather station to predict when storms, typhoons, and hurricanes will occur. He may simulate an entire manufacturing operation on the computer, searching for more economical or faster operating methods. He may set up a computer to calculate data that will be gathered from America's first manned satellite.

If you would like to work on problems like these and are a recent college graduate with at least two years of college mathematics, you may make an excellent Programmer. Many openings now exist. No previous knowledge of computer operation is necessary; IBM will provide comprehensive training. Salaries and future opportunities are excellent. From computer programming, you may move into a wide variety of career areas.

For details, write, outlining your background and interests, to:

**Mr. R. E. Rodgers, Dept. 36K6**  
**IBM Corporation**  
**590 Madison Avenue**  
**New York 22, New York**

**IBM**<sup>®</sup> INTERNATIONAL BUSINESS MACHINES CORPORATION

Figure 6-4: IBM's ad for attracting programmer trainees, 1959

business of IBM, still IBM salesmen received technical training about computers in IBM's sales training school in Poughkeepsie (e.g. Wyly, 2008: 45-8). IBM's SBC also became a major source of workforce for the software and services industry. Elliott (1964) reported that "almost everyone in the service field seems to have worked for [SBC] at one time or another.

In 1960, IBM expanded its efforts by opening its own 'graduate school' called the "Systems Research Institute" to train its personnel to use computer to find solutions to complex business and scientific problems. By September 1961, the institute had 2500 graduates. A report in *New York Times* described the jobs that graduates would find in IBM:

"The graduates of the institute are part of the "software" that a computer maker offers customers with the electronic "hardware". The systems engineer, as the graduates are called, becomes part of a team that includes a sales representative, who has responsibility for the account, and the customer engineer, who has responsibility for machine maintenance. As the third man on the team, the systems engineer analyzes the customers' request in the light of the general problems of the particular industry, and tries to tailor an automatic processor to the customer's real needs."<sup>13</sup>

IBM also ran a series of IBM conferences for its customers: these were organized as free-to-attend gatherings of individuals representing companies in an industry to discuss the issues regarding the uses of IBM equipment. Eric Weiss, a Sun Oil Company representative attending the conferences, remembered that at the beginning (in the 1950s) IBM used these classes to inform customers about potential uses of computers and thereby advertise its products. Later, on customers took a more central role presenting their own uses of computers, the challenges they faced and the like. Representatives from IBM's Applied Science department would take these into account when deliberating about future products (Weiss, 1988). None of the other vendors did reach such a level of integration between their sales, customer support and R&D activities (see Akera, 2002; for a review of other companies' efforts, see Fishman, 1981: 155-176).

While such an integrated approach to training of sales and programming staff seems to be a unique feature of IBM, the level of programmer training and support IBM provided is also significant in its own right. For instance, Martin Goetz (former programmer at Sperry Rand and co-founder of Applied Data Research) testified that (Fisher et al., 1984: 23):

---

<sup>13</sup> Johnsrud, J (1961) Computer makers set up own 'Universities', *New York Times*, 24 Sep 1961

*New  
opportunities  
for*  
**MATHEMATICIANS**

as  
Applied Science  
Representatives  
with

**IBM**

These are excellent positions requiring top level work as consultants and educators in the application of IBM electronic computing equipment to the most challenging problems of business and science.

The opportunities for growth in this vast new field are exciting.

Employment assignments are available in major cities throughout the United States.

Good salaries, excellent working conditions, exceptional employee benefits.

*Minimum requirements:* Major or graduate degree in Mathematics, Physics, or Engineering with Applied Mathematics equivalent.

*Desirable, but not required:* Previous experience in teaching applied mathematics and use of automatic computing equipment.

*Write, giving full details, including experience and education, to*

**Dr. C. C. Hurd, Director  
Applied Science Division  
Room 116**

**INTERNATIONAL BUSINESS MACHINES  
590 Madison Avenue  
New York 22, N.Y.**

Figure 6-5: IBM's ad for 'Applied Science Representatives', 1954



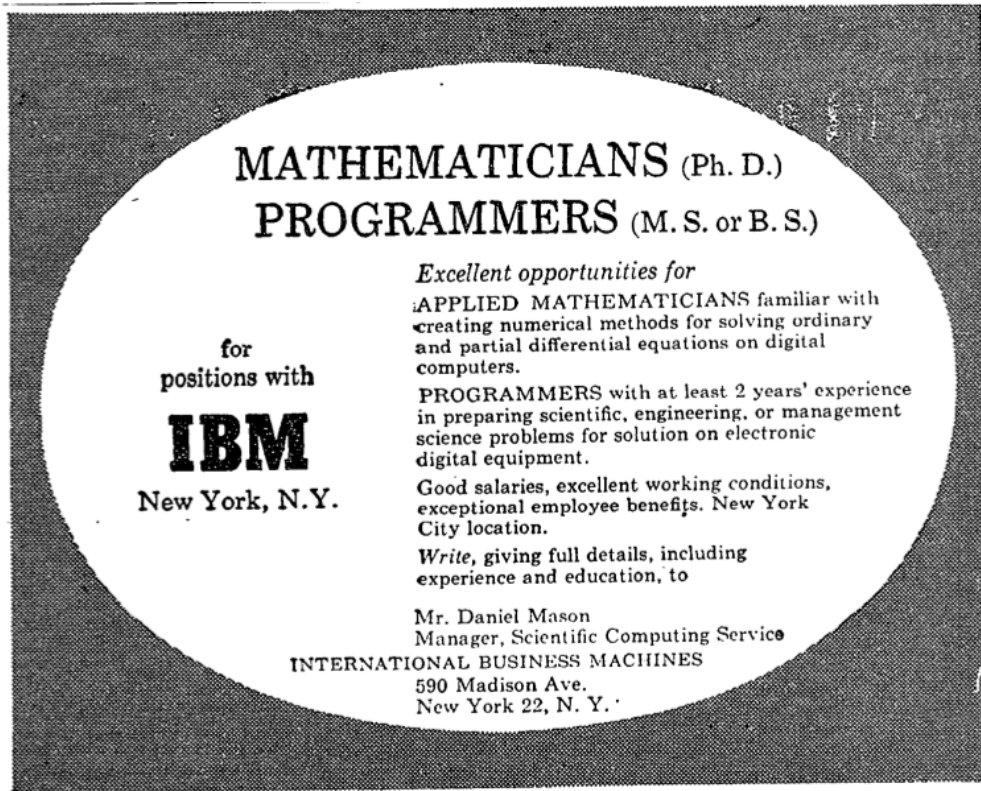


Figure 6-6: IBM's ad for programmers, 1954.

“Manufacturers made a concentrated effort to hire and train programmers as early as 1953. ...IBM acquired a reputation as a marketing-oriented firm which wouldn't desert a customer after a sale was finalized.... The capability for providing extensive “programming assistance”, therefore, became a significant criterion for evaluating competitive manufacturers' proposals.”

IBM continued to benefit throughout the 1960s. According to IBM's historian Pugh (1991: 638):

[T]he large reservoir of programming talent and experience in IBM's sales division was an important factor in the success of System/360. System engineers and programmers in the Data Processing Division not only performed critically essential services in installing the operating systems, but they also developed many general-purpose programs useful to customers.”

Other computer manufacturers provided training to their customers but the extent of their activity in the labor market was limited. Burroughs and Honeywell, for instance, did not provide public computer training courses until, respectively, 1963 and 1969.<sup>14</sup> The only

<sup>14</sup> Honeywell established a new organizational unit called Honeywell Institute of Information Sciences.

other vendor company that considered training as a significant part of its operation was CDC, which by 1969 owned 19 training schools called “Control Data Institute”. When CDC acquired C-E-I-R, it also benefited from acquiring the Automation Institute, which by 1967 had schools in 46 cities and claimed to have more than 100,000 graduates from its various computer-related courses (Elliott, 1967).<sup>15</sup> By acquiring C-E-I-R, CDC aimed at not only gaining access to an extensive range of skilled programmers, but also expanding its capability to train programmers and other computer-workers. In 1970 alone, more than 10,000 new students enrolled in CDC’s computer technician, programmer and operator courses. Moreover, some of the branches managed to get accreditation by local universities, most notably perhaps, from the University of Minnesota. Robert Price, CDC’s future CEO, later considered this acquisition “a pivotal” change for CDC’s data services (Price, 2005: 159, see also Robinson, 1988: 37).<sup>16</sup>

Another unique factor in IBM’s training was its determination to create a skilled workforce even beyond its employees. IBM had faced the limits that lack of trained programmers and system engineers placed on its production:

“It was as though we had the airplanes, but no one to fly them and no place to land...Sometimes we even found ourselves in a position where we had to hold back from taking a customer’s order”. (Watson and Petre, 1990: 261).

While the general feelings about the shortage of skilled computer workers were widespread, what was unique about IBM was its initiative to promote computer-training course in universities. According to Mauchly, as early as 1955, Hurd -IBM’s Director of Applied Science Division- was clear in his support for training by universities:

“[Hurd said] IBM recognizes the need for them to contribute funds towards educational programs in the computer field...[Hurd] went on to say that universities should be trusted to run their training in the best interests of all...He spoke against too much pressure from the industry for vocational courses and in favor of a broad liberal education”.

Mauchly concluded: “The main thing is to swell the number of persons who are not only active in the use of computers, but who in turn infect others with the possibilities of application and hence enlarge the computer market” (quoted in Fisher et al., 1984: 41-2).

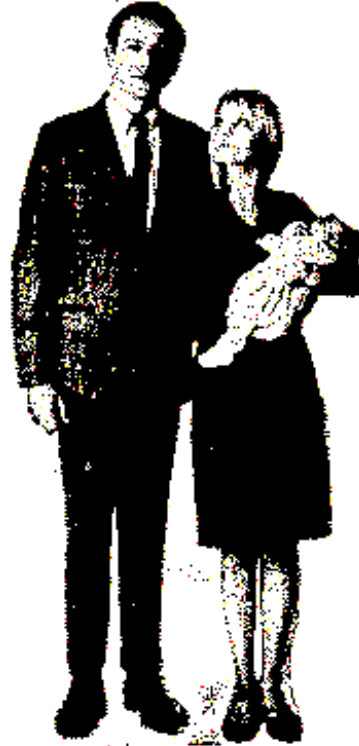
A report in *Datamation* (1959) found that apart from discounts, by the end of 1958, 50 universities had received IBM 650s as gifts. Watson Jr. considered IBM’s “very aggressive

---

<sup>15</sup> Elliott, R (1967) ‘THINKing big: in computer software the reach frequently exceeds the grasp’ *Barron’s National Business and Financial*, 2 Oct 1967;

<sup>16</sup> Other software companies that provided training operated at a much smaller scale. See the report on CAI and CUC in Elliott (1967).

We'll give you the opportunity  
to take a big step forward in life.  
Take it.



**Come Meet With Us**  
**10:00 a.m. Saturday, 16 March at**  
**Community Building in the Civic Center**  
**BELL AT MCKINNEY**  
**Denton, Texas**

In a matter of months, you could be well on your way to an easier life. You could be enjoying the job satisfaction, bright advancement potential, and financial reward of a career as a computer programmer.

And Control Data Institute can help you prepare yourself for it. Even if you've had no special training since high school.

We're the educational division of Control Data Corporation—the third largest computer manufacturer in the world. We offer courses in computer programming. And you learn on the Institute's own major computer installation.

If you would like an information packet on computer careers and the courses we offer, or would like to arrange for an interview and free aptitude test, call or write: Control Data Institute, suite

224, Garden Mall, Exchange Park, Dallas, Texas.  
Telephone: (214) 358-4271.

<b>CONTROL DATA CORPORATION</b>	<b>CONTROL DATA INSTITUTE</b>
Yes. Please mail me complete information on the exciting careers open now in the computer field, and details on the courses you offer.	
Name _____	
Address _____	
City _____	
State _____	Zip _____
Phone _____	
N 7-6	

Figure 6-7: CDC advertisement for programmer trainees, 1968.

college discount program” as one of the “IBM’s key moves” because it helped training of “a whole new generation of computer scientists who made it possible for the market to boom” (Watson and Petre, 1990: 261). Obviously IBM benefits from its gifts and discounts went beyond simple expansion of the industry. I have already mentioned the role of IBM’s educational discount on the popularity of FORTRAN. IBM used its influence in university computer centers to orient the future generation of computer workers and researchers. According to the Datamation study: “in many cases, to the extent that a university computer activity has a purpose at all, it has been made for them by IBM” (see also Fishman, 1981: 391-3).

#### **6.4. Rapid expansion of the labor market**

By 1967 IBM alone was spending between \$90 and \$100 million annually on training programmers, producing thousands of graduates every year (Ensmenger, 2010a: 71).<sup>17</sup> In spite of this, and all the efforts of other manufacturers, the shortage of software workers seemed to be worsening throughout the 1950s and 60s. It is difficult to provide an accurate size of the labor market because contemporary estimates are widely different. It is much easier, to give a sense of the scale of the problem that the industry and its users were feeling. According to a Datamation’s editors in 1962, “first on anyone’s checklist of professional problems” was “the manpower shortage of both trained and even untrained programmers, operators, logical designers and engineers in a variety of flavors.” Popular Science magazine, in a report containing a short illustrated guide to “programmers’ daily work”, stated that five hundred thousand men are wanted “to feed computers”.<sup>18</sup> In 1966, Business Week declared a “software crisis”: “Shortage of programmers -and the fruits of their solitary art- is stunting growth of computer use and costing industry hard cash.”

In such an environment poaching (by employers) and job-hopping (by employees) became the norm. The acute shortage of programmers was evident even at the very first session of IBM’s users group, Digital Computer Association: the inauguration was briefly disrupted by some of the participating managers who were angry at each other for stealing programmers (Campbell-Kelly, 2003: 32). Some employers even banned their employees from attending professional conferences because they feared that headhunters in those conferences might lure them away (Ensmenger, 2010a: 72). Even lower level trainees and novice programmers too could use changing jobs simply in pursuit of higher salaries. Richard “Dick” Brandon (a

---

<sup>17</sup> To put this in context, a survey of 300 colleges offering degrees in data processing and computer science had found that their total output in the academic year of 1966-7 was about 2800 (White, 1970).

White, T.C. (1970) ‘The 70’s: People’ *Datamation*, 15 Jul 1970.

<sup>18</sup> Englehardt, S.L. (1965) ‘Wanted: 500,00 men to feed computers’, *Popular Science*, Jan 1965

prolific speaker and commentator on the industry who was also engaged in training activities through his software service company) called the amount of increase the employers paid programmers every year to keep them “the loyalty gap” (Brandon, 1971: 28). According to Brandon, salary increases of 10 to 16 percent were normal. A report in the Fortune magazine in 1967 declared “Competition for programmers has driven salaries up so fast that programming has become probably the country’s highest paid technological occupation . . . . Even so, some companies can’t find experienced programmers at any price.”

**College Graduates**

**ARE YOU A COMPUTER PROGRAMMER AND DON'T KNOW IT?**

Many career-oriented people who have training in engineering, mathematics or one of the sciences are now going into computer programming. They find that it makes full use of their special analytical and logical talents, often not fully tapped in their present work.

In the 60's and beyond, the electronic computer will become more and more an important factor in the operations of business, industry, science and government. Computer programmers, the men who direct the computers, will play a vital role in this progress.

One of the most important qualifications for this work is the ability to analyze complex problems and to deduce from them meaningful and useful answers.

A computer programmer takes a problem — it may be in engineering, science, marketing or in any of dozens of other fields — analyzes it, and phrases it in a mathematical-logical language that the computer can “understand.” Then the computer goes to work with its prodigious speed and accuracy.

If you qualify as a computer programmer you will be given an intensive training course in machine language, stored program equipment, problem-solving techniques and program writing.

A degree in engineering, physical sciences, mathematics, or natural sciences is desirable with mathematics through differential equations preferred. All qualified applicants will be considered for employment without regard to race, creed, color or national origin.

**ALBANY INTERVIEWS**

Fri., Oct. 20, 5 p.m.-9 p.m.  
Sat., Oct. 21, 9 a.m.-7 p.m.

560 BROADWAY  
ALBANY, N.Y.  
TEL HO 5-1461

**IBM**

or write outlining briefly your background, to:  
Mr. C. T. Darrah Jr., Dept. 322K2  
IBM Corp.  
General Products Division  
Endicott, New York

INTERNATIONAL BUSINESS MACHINES CORPORATION

Figure 6-8: IBM’s ad for college graduate emphasizes that they may not know that they are programmers. 1961

Ensmenger (2010a: 71) argues that “[f]aced with a growing shortage of skilled programmers, employers were forced to expand their recruitment efforts and lower their hiring standards”. I have already documented the diverse backgrounds of the early generation of programmers that entered into the software labor market (in IBM, SDC and

elsewhere). The high salaries that employer were offering attracted many more people to the opportunities in the area of software work. By mid 1960s, there was no shortage of people who wanted to become programmers. Moreover, it had become clear that a high school degree is enough to get a job as a computer programmer. In fact, by one estimate, one in four programmers in 1965 were high-school graduates (Kiplinger magazine, 1965).<sup>19</sup> The report in Popular Science magazine (quoted above) had advised its readers that “[y]ou don’t have to be a college man to get a good job in programming- today even high-school grads are stepping into excellent jobs with big futures”. The doors of software work labor market were open, it was suggested, to all who were willing to take a chance.

In the “Mary Merrifield” column of Chicago Tribune on Oct. 3rd 1967, a girl named Shirley asked the following question:

“Dear Mary Merrifield: I am a high school senior and want to be a programmer for computers. I’m good at math. How do you suggest I go about this?”

Mary’s response (in full) was that:

“I am told by IBM and other training centers (for employes [sic] only), that a person who is good at math and has a logical mind, often is best for computers. However, there’s quite a difference between learning to program key punch cards and the more sophisticated type of programming. The best programmers are the ones who can see to the very core of the problem, and then anticipate all the variables. College and universities now often teach computer programming, so check the college of your choice. Also there are schools specializing in computer technology. You can learn the basic techniques in several months and then go from there. Check the school ads in The Tribune and discuss the field with your school counselor. As demands for computer programmers exceeds supply, good salaries and promotions are assured for those who develop skills.”

This can show vividly the extent to which discussions about programming skills and its relationship to jobs was trivialized. It would also demonstrate vendors’ implication in creating the situation. To take another example, in 1969 Honeywell started a data processing course specifically for training high-school graduates in “programming and computer-related systems analysis”. Students had to complete 475 hours through three-evenings-a-week classes that were held at Honeywell’s training facilities in Wellesley Hills (MA). Asked if the lack of a college degree would limit the opportunities available to course trainees, the program manager said: “Our experience is that a degree is not a requirement for success in data processing.” He nevertheless insisted that they are looking for “the

---

<sup>19</sup> Kiplinger magazine (1965) ‘Lots of new jobs in computers’, Aug 1965.

exceptional high school graduate[s]” and their aim was to provide them with “respectable EDP credentials”. He also mentioned that all applicants would be tested by company’s standard aptitude test.<sup>20</sup>

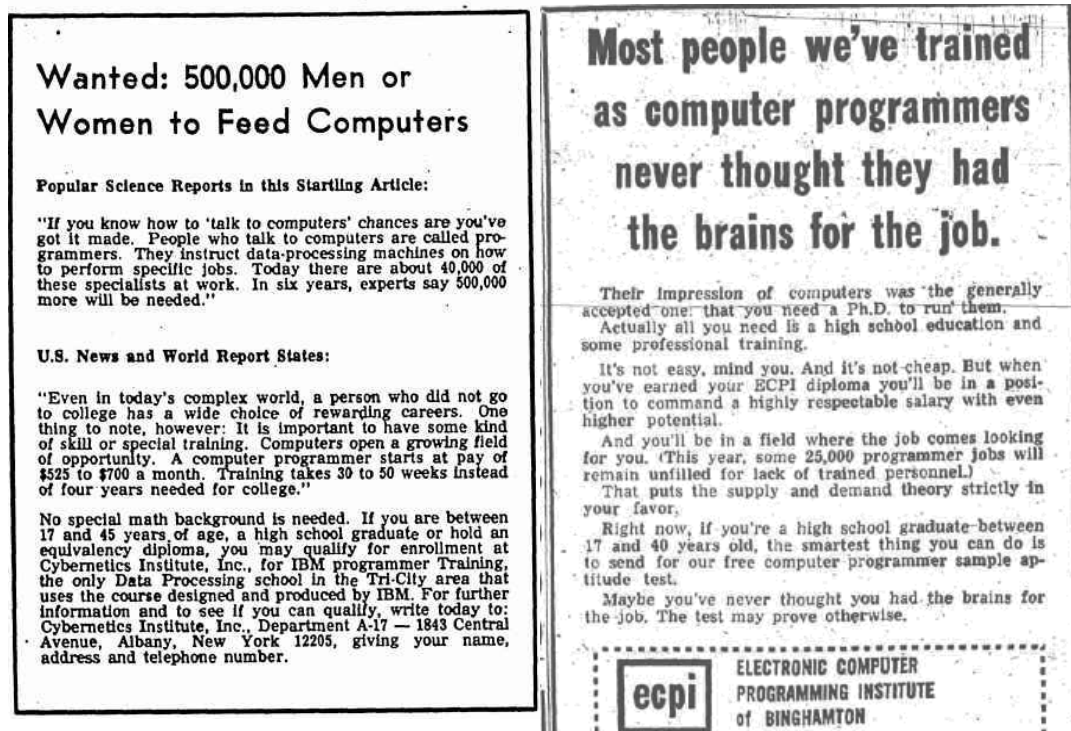


Figure 6-9: Two sample ads from EDP

schools in 1967

Given the amount of interest from the industry, their customers and the people who were seeking entry to computer jobs, the growth of independent training providers was predictable. In fact as early as 1957, training providers such as Electronic Computer Programming Institute and Automation Institute were established with many branches across the country. Such “EDP Schools” sprang up all around the country in the 1960s to train programmers. By one estimate, more than 700 EDP schools were founded between 1957 and 1969 and more were being founded almost every day.<sup>21</sup> Most of them were profit-oriented enterprises which had seen an opportunity to build a small fortune out of the muddled state of computer skills and the anxieties of the employers. Before long, their advertisements could be found everywhere, “from the classified section of newspapers to the back of paper matchbooks” (Ensmenger, 2010a: 74). They promised high salaries and excellent career opportunities and emphasized country’s desperate demand for programmers. Some went as far as guaranteeing placement upon graduation and were subsequently charged with “using

<sup>20</sup> Rachstein, K. (1969) ‘Honeywell adds DP course for high school grads’ *Computerworld* 24 Sep. 1959

<sup>21</sup> Menkaus, E.J. (1969) Electronic Data Processing Schools: Avenues to success or dead-end roads?, *Business Automation*, 27-29.

deceptive ads”.<sup>22</sup> In the eyes of one contemporary observer, it seemed that “the only meaningful entrance requirements” for these schools was “a high school diploma, 18 years of age...and the ability to pay” (Markham, 1968).

The training provided usually involved three to nine months of various courses which included key punch and tabulating machine operating, programming techniques and even basic arithmetic’s. Programming techniques were usually paper and pencil exercises rather than hand-on training, but some schools did provide limited access to computer to the students (usually through leasing from another company). Instructors were usually working programmer but very often without much experience. Depending on whether or not the school had access to computers and provided substantial training, the cost of a course was in the range of \$1,000 to \$2,500.

While the more legitimate training schools really wanted to contribute to the industry, unless they were organizationally linked to some vendor or service companies, they were most probably lost. There were very few textbooks and no established curricula. The only device that they were sure was useful was the aptitude tests. Consequently, training providers used these test in a variety of ways: in most cases some test (usually a “watered-down version of IBM PAT”) was used to give the candidate an indication of how suited he or she is to programming tasks (Ensmenger, 2010a: 76). But it was extremely unlikely for an institution to refuse admission to a trainee because of their aptitude scores. Some schools used test books and other material (included compromised tests from employers) as the basis of some of their courses. In the absence of any standard or regulation, training schools were effectively free to design their advertisement, entrance examinations, curriculum, and fee structure.<sup>23</sup> Indeed several institutions were later taken to the court by their students or graduates. By the end of decade some companies had established “no data processing school graduate” policy.<sup>24</sup> It is notable that while employers were complaining about the quality of graduates of EDP schools, some EDP schools were adamant that “the industry won’t spell out what type of training it expects programmers to have”.<sup>25</sup>

---

<sup>22</sup> Chicago Tribune (1972) Business Ticker Column, 3 May 1972

<sup>23</sup> The only regulation that they appeared to be observing was the GI Bill, which was designed to support war veterans.

<sup>24</sup> To be sure the provision of training (outside universities) went beyond these institutions. Community colleges and a number of high schools began to provide EDP training in this period. Also some (independent) publishers published self-study programming textbooks.

<sup>25</sup> Menkaus, E.J. (1969)



#### ***6.4.1. The socio-political environment of labor market***

The socio-political environment of the 1960s also affected the shaping of labor market, often stretching it into groups and communities that were neglected before.<sup>26</sup> The first influential trend was the rise of Civil Rights Movement and the heightened general conscience about racial issues. Some companies, like IBM who had adopted equal opportunity in the previous decade, used the opportunity to advertise its software and services jobs directly for non-white audience (like readers of Ebony journal). At the same time, activists within those communities became involved in raising awareness about the opportunities that were before barred by discrimination or lack of access. As early as 1963, Ebony magazine<sup>27</sup> included computer programming in the list of “professional, technical, white-collar fields” that are opening to non-whites. In July 1970, the same magazine<sup>28</sup> produced a list of 44 accredited EDP schools (in 16 states).<sup>29</sup> The report indicated that these were “highly paid” jobs that did not require degrees. “In-school” training, it was suggested, was enough to prepare individuals for the job market.

A second and related development was government’s Job Opportunities in the Business Sector (JOBS) program that encouraged employers to recruit from “the underprivileged”. Computer industry as a whole was active in participating in these schemes (and taking advantage of the benefits they provided), but perhaps the prime example of response to these initiatives was CDC, which between 1968 and 1974 almost tripled the number of minority (including female) employees in its professional and technical employees (Delton, 2009: 239-255). Other business associations and companies were also active. A group called Computer Personnel Development Association was set up to “secure openings in computer jobs for individuals from ghetto areas” (Burrows, 1969). They searched Local communities for programming talents and provided them with, among other things, computer operation and programming. After the Vocational Rehabilitation Act of 1968 entitled prisoners to vocational assessment, training and placement, various computer-training programs started in prisons across the US. IBM offered keypunch operator program at the New York State Reformatory for Woman and another prison in New Mexico and the Electronic Computer Programming Institute provided programming training for the inmates of the Sing-Sing Prison near New York. There were also computer-programming courses for the inmates of

---

<sup>26</sup> Some college and high-school graduates decided to enter computer jobs to evade drafting during the war in Vietnam. But it doesn’t seem to have affected the labor market significantly.

<sup>27</sup> Ebony Magazine (1963) ‘Job outlook for youth’, May, 1963

<sup>28</sup> Ebony Magazine (1970) ‘Careers without college’, Jul 1970.

<sup>29</sup> Accreditation was provided by either the Accreditation Commission for Business Schools or the National Association of Trading, and Technical Schools.

prisons in Atlanta (GA) and Jefferson City (MO) and data-processing programs in prisons of at least 5 states and District of Columbia (Bosworth, 2004 :1010).



**Programming at IBM**  
**“It’s a mixture of science and art”**

“A lot of people have the wrong idea about computers,” says Earl Wilson. “They think the machines solve problems all by themselves.”

A programmer at IBM, Earl got a B.A. in Modern Languages in June, 1967, and joined IBM a month later.

He’s now working on a teleprocessing system that will link computerized management information systems of several IBM divisions.

“When a computer comes off an assembly line,” he says, “it can’t solve meaningful problems until somebody writes a program—a set of instructions. And to do that, you’ve got to be part scientist, part artist.

“Science is involved,” says Earl, “because you have to analyze problems logically and objectively.

“But once you’ve made your analysis, you have to start thinking creatively. There’s a huge variety of ways to write a program, and the choice is up to you.”

Programmers hold a key position in the country’s fastest growing major industry—information processing. *Business Week* reports that the computer market is now expanding at about 20% a year, a rate many experts think will be sustained at least until 1975.

**You don’t need a technical degree**  
If you can think logically and like to solve problems, you could become an IBM programmer, no matter what your major. We’ll start you off with up to twenty-six weeks of classroom and practical training.

**Immediate openings for college graduates**  
If you have a college degree and are interested in a programming career at IBM, write to Corporate Recruiting Manager, IBM Corporation, Department BM1031, 425 Park Avenue, New York, New York 10022.

**An Equal Opportunity Employer**

**IBM** selected material

Figure 6-10: IBM’s programmer ad in Ebony, 1968

The government of United States took a direct interest in training software workers early in the 1960s. Since the federal government could not compete with the industry by raising

salaries and had difficulties in filling programmers and other computer workers, it made new provisions for training of computer workers. As part of the Vocational Education Act that was passed in the Congress in 1963 and considered government expenditure in training of “highly skilled technicians”, government was allowed to pay for computer training of its employees, “requiring in exchange only that they work for a certain time in government agencies” (Ensmenger, 2010a: 72). By 1966, government had paid for training of 33,000 computer workers. But beyond this Act, government did not take any action to regulate the market.

#### ***6.4.2. Associations of software workers***

At the same time that most of the employers (including the large vendors) did not take any interest in restricting entry to the computer labor market, there was much discussion about the ‘emerging profession’ of computing (or data processing) within the academic circles and some of the trade journals. But most of these discussions were concentrated on the quality of programmers work, their relationships in the workplace or even their code of ethics. Few considered the possibility of regulating entry to the computer market. ‘Professional’ institutions like ACM were more concerned about the scientific respectability of their organization and their members than creating a common ground for all of the software workers. They looked down on other, less educated, groups of programmers and systems analysts, especially those who were engaged in non-scientific - that is, business-applications. It was hardly surprising then, that when the 1968 guidelines for computer science programs were published they had virtually neither binding power, nor any support from the other groups. Notwithstanding the overtly abstract and theoretical orientation of the proposed curriculum, the limited reach of computer science departments within the labor market made it almost impossible for the recommendations to have any effect.

The practitioners in businesses, on the other hand, were far from united in their structure or demands.<sup>30</sup> The largest occupational association of data processing workers (the Data Processing Management Association, DPMA) was the outgrowth of an association of accountants and tabulating machine managers (called National Machine Accountants Association, NMAA). In 1952 it represented more than 10,000 data processing personnel, but probably most of the members were not computer workers. By 1962, when probably most of the members were in computer-related jobs, its 16,000 members constituted only a small fraction of the computer workforce. Nevertheless, DPMA tried to represent all sectors

---

<sup>30</sup> This section draws largely on Ensmenger (2010a: Ch.7), for an official history, see Coha, 1982).

of the software work in its membership and, at the same time, define limits for entry of individuals to the labor market.

DPMA was the main association that had some success in establishing a knowledge-based exam and certification for practitioners. According to DPMA's executive director in 1963, one of the main objectives of such an effort was to stop the flood of amateur programmers and to roll back "bogus" data-processing schools. DPMA's main strategy for imposing such limits was its Certified Data Professional (CDP) exams that started from 1962. The exam itself was a legacy of NMAA but was introduced in the same year that NMAA's name was changed to DPMA, thus highlighting the new orientation of the association. The purpose of exam was to examine candidates' knowledge of the data processing field without emphasizing a special educational background. Sitting the exam did not require membership in the association, but the candidate had to graduate from a prescribed 3- year college program and have 3 year of related experience.

DPMA wished that the exam would set "a standard of knowledge for organizing, analyzing and solving problems for which data processing equipment is especially suitable." But when the exam was first taken in 1962, it was widely criticized for its lack of depth and even irrelevance to real-world problems. Beyond programming, the 1962 test covered such subjects as Boolean algebra, numerical analysis and elementary cost accounting and even basic mathematics and English. It also specified a list of certain academic courses that candidates should have had passed as a prerequisite for sitting the exam. Many DPMA members found this requirement "irrelevant, unattainable, or both" (Ensmenger, 2010a: 183). Soon the requirements were changed to a two-year college certificate but it still remained controversial.

Despite the controversies, the exam enjoyed some level of initial success, especially because, in response to criticisms, DPMA dropped its educational requirements for three years. In 1965, a record number of candidates (nearly 7000) took the exam and about 5000 passed. More importantly perhaps, several large employers (like the Prudential Insurance Company of America and the US Army Corps of Engineers) officially recognized the certificate. Given that DPMA was pioneering this filtering method in an otherwise unregulated labor market, their moderate acceptance in the industry could be considered a relative success.

But the initial success was short-lived, partly due to the lack of support from other associations and partly due to the inadequacies of DPMA. ACM, for its part, used every

opportunity to undermine DPMA's position, from issuing an statement to warn employers "against relying on examinations designed for subprofessionals or professionals as providing an index of professional competence", to objecting to the notion that CDP stands for "Certified Data Professional". Under pressure, and perhaps concerned about the legal implications of its certificate, DPMA removed all references to professional status from its exam and related material and CDP came to stand for "Certified Data Processor". It also conceded, in a statement, that CDP does not provide "assurance of competence" and is not intended as "a requirement for employment or promotion in the field" (all quoted in Ensmenger, 2010a: 183).

At the same time, there were complaints about incompetent administration of exam and outright accusations of fraud. DPMA itself, which provided CDP-related courses in its local branches, was accused of business-like operation. Soon DPMA found itself incapable of even enforcing legal use of its trademark exam and certification. Therefore, with the passage of years, rather than gaining momentum, CDP lost momentum. The number of candidates dropped and stayed below 3000 (sometimes much lower). It took ten years for the total number of candidates to reach above 31000 (by which time the number of CDPs had reached 15,000). The numbers are so small that compared to the number of PAT tests administered, they seem ignorable. But the fact that CDP was the best-known and most widely recognized professional certificate in the 1960s tells much about the frictions and misalignments among software workers.

ACM and DPMA, which were perhaps the most powerful bodies, were nevertheless only two of the numerous 'professional' associations that existed. Each of these associations had different backgrounds and histories and different preferences. Most importantly, computer workers and their various associations did not reach any agreement about what constituted a standard of knowledge that one had to have to work in the industry. As long as this schism continued, there was no way to exert any influence over the further expansion of labor market, let alone establish a profession. As one perceptive contemporary observer commented: "[t]he manner in which people enter these occupations bears no resemblance to the way in which candidates gain admission to the autonomous professions, viz. By long training rigorously prescribed by the profession itself and sanctioned by the State" (Rhee, 1968: 117-8).

## **6.5. Employment policies**

Most of the companies that entered the early computer industry pre-dated the industry. Put differently, despite the pioneering efforts of Eckert and Mauchly, it was entrance of those long-established companies that shaped the emergent industry. In doing so, they carried over their employment practices to the computer industry (Haigh, 2010). Since many of these practices were shaped in during the Second World War, I start by considering the shaping of employment relationships during the war and then consider how it developed in the special cases of individual companies.

Jacoby (2008) provides an excellent picture of the post-war labor market and corporate employment policies in the US. According to Jacoby, following the years of depression and recession in the 1930, US government had become heavily involved in the economy and especially regulating the labor market. Unions were provided with legal backing and employment security and other benefits had been put in place for large proportions of the population. The Second World War brought government closer to businesses as part of the massive industrial effort that was required for the war. On the other hand, this mobilization of country's resources and the burden of war meant that companies faced risks of shortage of workforce. Therefore they tried to rationalize their procedures and role structures without alienating workers. At the same time union membership grew rapidly within the ranks of workers.

As part of their efforts to cope with labor shortage problem, the industry initiated massive training programs during the war. The standard procedure for developing those programs involved projecting labor requirements, simplifying jobs, and training by progression from one job to another. At the same time, the US government which had become even more involved in regulating the market, started to set standardized employment conditions across the industries. This included requiring all firms to "describe their jobs in standard terms taken from the recently compiled Dictionary of Occupational Titles. This requirement forced firms to conduct extensive job analyses and to create a rational structure of well-defined but rigid job titles." (P. 196) It also supported most of the personnel practices that leading corporations had established, including centralization of recruiting and hiring decisions (which were previously made at shop floor level) and standardizing pay structures.

As the end of the war came in sight, many workers were fearful that after the war they will lose their securities and a recession period like the 1930s would occur. Employers, on the other hand, did not want to give in to increase union demands. The post-war Republican government, for its part, adopted anti-union policies that restricted unions' political activities

and made industrial action more difficult. As a result unions became less militant and looked for ways to insure the economic interests of their members. At about the same time, several management consultants advocated the view that “to counter union demands for guarantee plans, management would have to try harder to stabilize employment“ (p. 199). Subsequently many companies adopted such policies and started proving worker-friendly policies like job security, paid vacations, increased benefits and even corporate-sponsored social clubs.

IBM and NCR were two of the companies that had championed such policies of “welfare capitalism” even before the war. These companies had sought “to bridge class barriers by creating ‘bonds of shared belief, ethnicity, and gender’” so that workers felt themselves part of industrial communities in which they aspired to rise through the corporate ranks. Profit sharing and the creation of internal labor markets<sup>31</sup> bound workers more closely to firms” (Haigh, 2010: 11). IBM perhaps was the more famous of the two because of the distinct corporate culture that Watson Sr. had created during his time as chairman and his efforts to publicize it (until 1952). Some of the more unusual elements of corporate culture were carnivals and picnics for thousands, corporate songbooks and firework celebrations. IBM’s choice of its plant locations also magnified the distinctiveness of its internal culture: IBM always preferred previously unindustrialized areas outside large cities. This ensured that immediately after opening the plant IBM would become the largest employer in the area. The paternalistic culture inside IBM and the geographically bounded community that lived around it gave a particular cohesive appearance to the way IBM managed its workforce.

More substantively in terms of employment policies, it was claimed that IBM had not laid off any employee since 1921 even though IBM did not have an explicit policy regarding employment security.<sup>32</sup> Moreover, IBM provided employment security for all levels of employees, encompassing blue-collar, technical/professional and managerial levels. This had led to relative blurring of blue-collar and white-collar jobs. More importantly, IBM offered lower-level employees the chance to rise to professional positions through its various promotion mechanisms and training programs. Many IBM employees went through “earn-as-you-learn” training courses and in some cases company paid for employees’ college education (Loseby, 1992:118-123; Mills, 1988). Watson Jr wrote in his memoir (Watson and

---

<sup>31</sup> The corporate practice in which existing employees have priority in selection and assignment for new positions.

<sup>32</sup> At the shop floor level, this was primarily achieved through HR planning, understaffing (keeping permanent employees at about 85% of the level required) and use of temporary workers and subcontractors at times of extra demand

Petre, 1990:332) that he does not believe that “[his father’s] primary motive was to keep the unions, but that was one effect”.

Thus when IBM entered the computer industry in 1952 it employed more than 41,000 employees. At the time, IBM not only produced typewriters and cardpunch machines, but also time measurement equipment and military products. Thanks to the SAGE contract and opening of new lines of production (in “electronic data processing”) IBM expanded throughout the decade. By 1958, it employed nearly 87,000 people. During this period, even though IBM had sold its time equipment division and converted its lines of military products, affected IBM employees were retained, retrained and assigned to new jobs.

The importance of maintaining this policy for senior managers can be seen in the two following cases which both are related to IBM’s software workers. In 1954, when IBM turned down accepting responsibility for programming of SAGE, one of the reasons was because it was afraid of its subsequent employment commitments. Robert Crago, manager of SAGE program at IBM, later remembered that “...[W]e couldn’t imagine where we could absorb two thousand programmers at IBM when this job would be over some day” (in Tropp, 1983: 386).<sup>33</sup> Two years later, the 1956 consent decree specified that IBM and SBC couldn’t employ the same employee. According to IBM’s historian Emerson Pugh, this created major problems for IBM because of its implications for its employees: they had to either “leave the IBM or find entirely new assignments” (Pugh, 1995: 255 and 381n.2).<sup>34</sup>

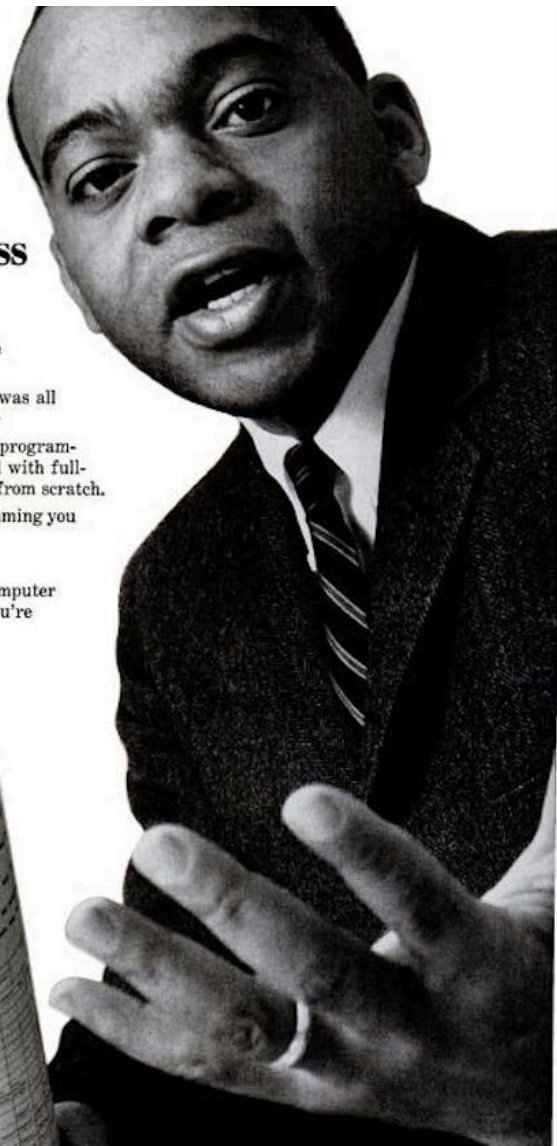
By 1969, when IBM employed more than 250,000 employees, it was not only the leader in the computer industry, but also a show-case example of corporate America in which not only militant unions had no place but amicable relationships between managers, workers and employees and stable employment relationships created social harmony. In this sense, IBM was a prominent but by no means the only example. Most of the major vendors seemed to share and represent the same values and progressive line of thinking. While NCR, GE and DEC are known for their support for employment security, there is not much available about the employment policies of others in this period. Nevertheless, their job adverts appealed to the same ideals and made similar promises. It is safe to assume that the 15-year-long boom of the computer industry in this period, did not test the commitment of these companies to their employees and, given the widespread shortage of skilled workers, they all adopted policies which were similar to IBM -the industry leader- to attract individuals.

---

<sup>33</sup> At the peak of its engagement in SAGE, IBM had 7000-8000 employees involved in that project (Flamm, 1988: 89).

<sup>34</sup> Later on, when IBM sold SBC to CDC, it paid CDC, among other items, \$26 million in fringe benefits of employees (more than what it was receiving for the division).





**“I think you can measure  
a company’s interest in  
its people by its willingness  
to invest in them.”**

“When I interviewed IBM, the tests showed I had a high aptitude for computer programming.

“I was glad to hear that, but I only had a foggy idea of what it was all about. (This is Alvin Palmer, an Associate Programmer at IBM.)

“IBM soon took care of that. They sent me to one of their major programming schools. This isn’t on-the-job training, but an actual school with full-time classes. You really learn how to be a programmer, starting from scratch.

“The school lasts up to 26 weeks, depending on the kind of programming you go into. And IBM pays your full salary the whole time.”

**You don’t need a technical degree**

“I get a tremendous kick out of programming. You’re telling a computer how to do its job, and it really gets you involved. Maybe because you’re continually solving problems. Often very complex ones.

“But you don’t need a technical background like mine. There are plenty of programmers with degrees in liberal arts or business. What counts is having a logical mind.

“I’m making good progress in this field, so I’m glad IBM invested in me. I think it indicates how far they’ll go to help you make the most of your abilities.”

**Immediate openings for  
college graduates**

Al’s comments cover only a small part of the IBM story. If you’re a college graduate and would like more information, send an outline of your career interests and educational background to C. J. Reiger, IBM Corporation, Dept. NA8-P, 100 South Wacker Drive, Chicago, Illinois 60606. We’re an equal opportunity employer.

**IBM**

Figure 6-11: IBM’s programmer ad in Ebony, 1968

Still, not all of the history of this period is bright in terms of employment relationships. There is some anecdotal evidence that shows Remington Rand managers, before they joined Sperry, were much more hostile in their employment relationships. Remington Rand’s president, James Rand, was notorious for crushing a union strike which had led to violence and firing of all striking workers in 1936-7. Herbert Mitchell, a distinguished programmer and the first doctoral student to graduate from Howard Aiken’s laboratory, wrote in his memoir about the attitudes of Remington Rand towards workers:

“Not long after Remington-Rand acquired Eckert-Mauchly, the employees decided to form an association, a euphemism for union. Some of the leaders urged me to become vice-president. I was favorably considering it when Art Draper [James Rand’s assistant] took me in hand and pointed out that I had to decide whether I was part of management or part of labor — I couldn’t be both. That was rather a shock to me, but I realized that he was right, and declined the position.<sup>35</sup>

Nevertheless, the union of manufacturing workers was created and had some power. A few years later, when James Rand was not happy with the pay scale of existing manufacturing workers, he asked Bill Norris (who was in charge of the ‘Univac’ division) to reduce it. According to Norris when the union rejected the new proposal, the senior management told Norris that they were willing to bribe the union leader and they expected him to get the deal (Norris, 1986). Norris took a stand against the managers, and given the already troubled relationships, quitted soon after.

The situation in programming and services companies was different from large corporations. On the one hand, their survival depended on having access to a group of skilled programmers who would be able to satisfy the terms of contract when a new contract would be signed. On the other hand, it was extremely difficult to keep programmers if there were no projects to keep them busy. In the context of programmer shortage mentioned above and the securities that work in large corporations provided, it was unlikely that salaries or even profit-sharing schemes were effective. So programming and services companies had to find ways of retaining programmers (especially those who could bring contracts or those with substantial experience) without making financial commitments. In the second half of 1960s, many of successful software companies (including those mentioned above) became public and their shares soaring over relatively few years. Their most common strategy for keeping programmers was offering them shares in the company. But even so, there was nothing preventing experienced programmers from leaving if they thought they could be independently successful and many did think so.

## **6.6. Implications for software work**

Throughout the first 15 years of computer industry, the market was expanding. The main segment of market was around the vendors that provided total solutions including mainframe, its peripherals and related software and services. The niche markets of minicomputers and independent software and service firms were too small to have an impact

---

<sup>35</sup> Mitchell had been working as a ‘printer’ while a student and was a member of printers’ union up to 1950.

on the overall competition. IBM was the dominant vendor in the industry and other companies had to define and defend their markets in relation to IBM.

While IBM's success in taking the leadership in the market could be dependent on many factors, its ability to maintain leadership was to a large extent due to its extensive training programs and integration of sales and marketing and support services. Historians generally agree with the analysts of the time that "[o]f all the computer manufacturers, IBM provided by far the most comprehensive programmer support and training [...]" (Campbell-Kelly, 2003: 30). At the same time, IBM treated its employees to some of the best, if not the best, salary and benefit packages. Moreover, IBM's flexible career lines and internal training programs allowed employees to grow and rise inside the company. Other companies tried to follow the lead of IBM but could not match it mainly because they did not have the resources (and the growth) that IBM had.

On the other hand, throughout this period, the question of skills were mooted and replaced by the more pragmatic issue of selecting among the available candidates. Aptitude tests became the most common and central element in selection and hiring of programmers and other computer workers. As demand for programmers grew day by day and the fears of an imminent shortage problem loomed larger than before, salaries spiralled out of normal limits and everybody gravitated towards taking up programming jobs. Vendors, their customers and training providers, each for their own part, expanded the limits of who could be eligible for software work to the extreme. The government, which had no interest in regulating the labor market, started its own training programs. Poaching and job-hopping became common practices and the minimum required education for entering the computer labor market reached the average American high-school diploma.

The various associations of software workers that emerged during this period failed to overcome their differences and take a united front. While they all aspired to the idea of professionalism and claimed to act in the interests of profession, their understanding of what it meant to be a profession and how it could be achieved could hardly be more divergent. Whenever the question of necessary skills and knowledge surfaced, there was little common ground between different groups. Most importantly, their schism meant that none of them had any power to influence the labor market in a tangible way. They were competing against each other. Meanwhile, the vendors in the industry were approaching the issue of skills from the standpoint of their commercial interests. I mentioned the motivations behind industry's efforts to move towards compatibility. Inside IBM, however, there were other important

factors considered in the decision: compatibility could save IBM tens of millions of dollars in terms of programming, training and support costs.

“Commonality in processor logic and programming was anticipated to provide IBM with economies in training of field personnel, development of programming, and standardization of installation and maintenance procedures.... The training of programmers, salesmen and systems engineers was made considerably easier because they had to be trained for one group of machines instead of for different, incompatible machines” (Fisher, et al, 1984: 113).

This prospect of rationalization and economization of training and services was a driving force behind IBM’s move. However, as far as the software effort is concerned, it is hard to argue that the anticipated economies were achieved. Indeed, the whole development project has been described as “little short of fiasco” (Campbell-Kelly et al, 2013: 131). OS/360, the operating system of System/360 computers, engaged more than a thousand software workers for four years and cost \$100 million<sup>36</sup> but was delivered 9 month late and riddled with problems. Its fate was saved by IBM’s more conservative strategy: in order to avoid a complete break from previous systems and facilitate transition of customers from pre-System/360 to System/360 computers, IBM had developed emulators which allowed customers to use the software developed for their previous computers on the new ones (Flamm, 1988: 99). This meant that System/360 had no discernible immediate impact on programming efforts of companies or services of IBM. More importantly, the common strategy of providing free training, software and support for customers did not change (Fisher et al., 1984: 174-5).

But even without System/360, the 1950s and 1960s were periods of rapid and frequent technological change: the switch from analogue to digital, the introduction of transistorized processors, the use of phone lines as communication networks, all occurred within this decade. Nevertheless, as the seamless integration of ENIAC girls and Grace Hopper into their programming roles at EMCC demonstrates (see Beyer, 2009: 191-9), the relatively small occupational community of programmers in the 1950s had too much in common to allow their differences fragment them. But as the labor market grew and the unchecked entry of individuals into labor market continued, the common bounds among software workers became thinner and thinner. Just as there was no common understanding of the required

---

<sup>36</sup> Before OS/360, IBM spent 10 million a year on software (Fishman, 1981 :96).

knowledge and skill, there was no common grievance to form boundaries around software workers.

Within this context, the employment policies that provided employment security and opportunities for growth within large vendors were a measure of stability in the labor market. Even though these policies were implemented by utilizing flexible career paths and facilitated lateral movements between departments, the training that companies like IBM provided to prepare their employees for new assignments became a common ground for the employees of those companies. Nevertheless, as we saw in the case of several software companies, some founded by former IBM employees, attrition and voluntary separation were not unseen. As long as the structure of industry remained unchanged, it was unlikely that the jobs would change in any significant way. But industry was about to change...

## **7 An IBM world: 1969-1985**

### **7.1. Competition in the industry**

In the late 1960s, a number of companies (mostly manufacturers of plug-compatible parts for IBM computers) filed lawsuits against IBM, accusing it of monopolistic behavior. The most high-profile of these lawsuits was filed by CDC in December 1968. In preparing its case against IBM, CDC collaborated with US Department of Justice. The Justice Department started investigating IBM operations in 1967 but had not yet decided to take action against IBM when CDC filed its lawsuit. IBM -mindful of the potentially disruptive consequences of a full-scale trial- attempted to “mollify” Department of Justice’s action (Watson and Petre, 1990: 406) by voluntarily terminating its policy of bundling of software and services with hardware sale. According to the new policy, announced in December 1968, IBM would start charging for its software and services. Nevertheless, the Justice Department filed its antitrust lawsuit against IBM in January 1969.

IBM announced details of its “separate pricing policy” in June, 1969. The announcement showed that, rather than an indiscriminate approach, IBM had selected specific parts of its services and some software products to be charged separately. In doing so, IBM’s objective was to:

- 1) Open up competition in parts of the software and services (in order to convince the Justice Department and market that IBM was serious in unbridling),
- 2) But keep those parts of bundled services that would be necessary for profitable sales (see Grad, 2002).

IBM decided that “system control programs” (like OS/360) were an integral part of the functioning of the hardware and could not be separately charged. While this was a technically defensible decision, there could be other motivations as well. On the one hand, IBM had spent so much on OS/360 that if it was to be charged separately, just to recover its development costs, it had to be offered at a much higher price than what customers were willing to pay. On the other hand, if OS/360 was unbundled, competition in operation systems was likely to draw away customers not only from using OS/360 but also all other IBM-provided software. By keeping OS bundled, IBM could contain competition and ensure that a large section of its customers will continue to use its services. On the other hand, IBM could point out that it had opened up competition in a wide variety of software products including sorting programs, conversion aids, language processors, file-management systems, etc. Even so, IBM’s leasing strategy continued for the software: customers had to pay annual rentals to be ‘licensed’ to use IBM software.

IBM also started charging for its “field engineering” services which meant that debugging and troubleshooting of most IBM software products could now be separately charged. Beyond software products, IBM started charging for all its custom programming services, which IBM called “systems engineering” services.<sup>1</sup> IBM programming services could be contracted on a cost-plus basis or, in a competitive bidding process, for a fixed-price. Therefore, all customized software development services that IBM used to provide for free were now charged (unless they had been planned before the announcement). Finally, IBM started to charge for all its training and educational courses with the exception of those that were aimed at introducing IBM’s products and services or exchanging customers’ experiences (like IBM conferences).

Even though speculations about the details of IBM decision and its consequences had started early on, the announcement of this separate pricing plan caused a stir in the market. Many expected IBM to drastically reduce its hardware prices (estimates varied between 25 and 40%). But when IBM reduced its prices only by an average of 3 percent, it was viewed by many as a covert way of increasing the net price of software and hardware. This gave IBM’s competitors a chance to boost their revenues by increasing their prices and charging separately for their software (if they wanted. For details, see Head, 1971: 9-10). Univac, Honeywell and RCA decided not to unbundle their products and services. GE unbundled its software and services only in its industrial process control systems. Burroughs had already started charging for some of its software products before IBM’s announcement in June, but did not unbundle its training services.

The only mainframe vendor company that changed significantly following the unbundling decision was CDC. CDC increased the price of its hardware by 5% and unbundled its software and services from hardware sale. Only a minimal software package (including an operating system and a business or scientific compiler or both) was provided free with every installation. Moreover, CDC reorganized all its service training, maintenance, data and programming services under a new organizational unit known as the “Services Group”, or more officially, the “Professional Services Division” (Price, 2005: 29). In 1970, this group had more than 1000 consultants, analysts and programmers that provided managerial consultation services as well as systems analysis and programming support and contractual programming services. There were still no charges for pre-sale consultation which was

---

<sup>1</sup> Systems engineering services “included a range of customer activities, including systems analysis and design, preparation of application flowcharts and block diagrams, program writing, and installation planning”.

provided by technically skilled software workers. In a manner very similar to that of IBM, these software workers were part of the marketing and sales organization providing advice on how CDC software could be useful for customers, but also closely linked to CDC's central software development team. Nevertheless, CDC's concentration remained on providing data processing services.

The results of these changes were augmented by the outcome of CDC's private lawsuit against IBM. In its private lawsuit, CDC had accused IBM of damaging CDC's sale by predatory pricing and premature announcement of a line of superfast computers. After several years of legal wrangling and negotiation, the case was settled out of court in 1973. In the settlement, among other things, IBM agreed to sell its Service Bureau Corporation (SBC) to CDC at its book value (much lower than its actual business value) and to stay out of the data services market in the United States for 6 years. After acquiring SBC, the number of CDC data centers grew from 27 to 71. Moreover, according to the settlement, none of 1700 workers of SBC could join IBM until three years later. This put CDC in an advantageous position in the data services market and moved data services to the center of CDC's competitive strategy (Price, 2005: 29-30).

The more important and long-term consequence of IBM's unbridling decision was an increase in the amount of interest and activity in software products. Unbundling of software products gave credibility to the claims of many software entrepreneurs who, for years, had pioneered the idea that software could be sold as a standardized product on the market. In the late 1968, viability of their idea was undermined by a decision by the US Patent Office that ruled out patenting of software products. While other means of protecting software as intellectual property were still available, software companies insisted that they needed full protection of the law in order to be able to prosper. The arguments for and against patenting software carried on for another four years. The decision was not settled until 1972, when the Supreme Court of United States ruled against patenting software<sup>2</sup>. In the process, the computer industry was divided between computer vendor companies (including IBM) - which generally opposed patenting of software- and software companies promoting it. Nevertheless, the net effect of unbundling of software by IBM and discussions around software patentability was that the software industry became much more visible.

Still, in terms of actual growth, the immediate effects of unbundling on software industry remained small. On the one hand, as mentioned before, even before unbundling decision a

---

<sup>2</sup> The legal decision had specific qualifications and was later partly reversed by the same court. See Campbell-Kelly (2005) and Lee (2014).



huge number of start-up companies had appeared in the software industry. On the other hand, few companies in the industry had enough experience and capital to produce and market software products. This was combined with the economic recession of late 1969 and early 70s which affected the whole computer industry. Therefore, while the sale of software products grew in the immediate aftermath of unbundling, the number of software and service companies almost halved and those who survived were struggling (see Campbell-Kelly, 2003: 82-7 and 118-19).

Early 1970s were, in fact, the first challenging economic period that the computer industry experienced. During this period, GE and RCA decided to leave the industry. Honeywell bought GE's computing operations in 1970 and RCA sold its computer division to Sperry in 1972. Overall, computer deliveries in the first two years of 1970s fell by 20 percent and even IBM's revenue stalled. It was not until 1973 that the industry returned to growth. But beyond a general decline and recovery, the structure of industry remained quite familiar: IBM had by far the largest share of the market and the BUNCH (Burroughs, Univac, NCR, CDC and Honeywell) were following it. Some new companies entered the market, most notably Amdahl Corporation (founded by Gene Amdahl, one of the designers of System/360) and Cray Research (founded by Seymour Cray, former chief designer at CDC). New companies had also entered the minicomputer market but the market share of these companies was small and DEC remained the dominant minicomputer vendor. It also expanded its operations to produce mainframe computers.

Perhaps the most significant development of 1970s was the shaping of corporate software products industry. While services in programming, data processing and facilities management were established sectors of the software and services industry, software products, as mentioned above, started to grow "steadily but undramatically" throughout the decade after IBM's unbundling decision (Campbell-Kelly, 2003: 121). These products proved to be much like traditional "capital goods" (like the computers they were run on) with high marketing expenses, extensive need for support before and after sale and ongoing maintenance costs. This led to what Campbell-Kelly calls "IBM-style" customer account management, technically-trained sales force, product documentation, user training and after-sale services.

The major suppliers of corporate software products were computer manufacturers (of mainframes and minis), independent software companies and turnkey vendors.<sup>3</sup> Computer manufacturers provided system software as well as applications, increasingly without any on-site technical or educational support. Customers usually had little choice in system software (like operation system) but at least could compare the combination of hardware and software options provided by alternative vendors in a more informed way. At the same time, building on the legacy of their user groups but changing their free-exchange logic, computer manufacturers started programs aimed at appropriating and selling software that was developed by their customers (like IBM's Installed User Program and Honeywell's Industry Application Acquisition initiative).<sup>4</sup>

Independent software companies were more focused on applications, occasionally making inroads into programming tools or utilities. Most of these companies were founded in the previous two decades, but a few joined them in the 1970s that went on to become powerful companies. ASK Computer Systems, Computer Associates and Oracle were all founded in this period. Turnkey suppliers were in fact hybrid hardware-software companies providing specific combinations of hardware and software tailored to the needs of customers in particular industries. For example, SDC, which had become a for-profit corporation in 1969, quickly diversified into many areas of activity. Beyond offering programming, data processing and facilities management services, SDC provided a turnkey solution called TEXT II for automating operations of newspaper companies. TEXT II comprised a configured network of computers and application software.

In mid 1970s, when unbundling had finally become the norm in the market, by far the largest share of software products belonged to IBM (40-45%). IBM's dominant position was maintained throughout the 1970s (and into 80s) despite new entries into the industry and growth in product sales. Towards the end of the decade, when the total sale of independent software industry had an estimated value of \$2 billion, IBM software revenue was estimated to be between \$400 and \$ 600 million. Moreover, while more than 6000 independent software companies claimed to be active in the software products industry, fewer than a dozen had revenues of more than \$10 million. 940 companies accounted for 68% of independent software sales.<sup>5</sup> As in the previous decades, software companies typically started as small, local businesses and a great majority of those that endured remained small throughout their lives. Their survival was largely thanks to a single successful product and a

---

<sup>3</sup> Two new forms of software service (brokerage and time-sharing) emerged in this period but were largely unsuccessful.

<sup>4</sup> *Computerworld* (1979) 'Honeywell backs user-built software', *Computerworld*, 2 Oct 1974.

<sup>5</sup> Estimates are from Frank (1979) and Campbell-Kelly (2003: 125-7).

small number of loyal customers. Many simply faded away when their founding members approached retirement. The more ambitious companies tried to reach a wider customer base and provide a variety of products. Some were successful but many were not. As the 1980s approached, in a frenzy of mergers and acquisitions, many of them joined together or became a subsidiary of computer manufacturers.

As a result of competition from leasing companies and plug-compatible manufacturers, IBM had seen its profit margins decline and started to look for ways of staying ahead of competition. During the early 1970s, it gradually entered the minicomputer sector by producing minicomputers for specific segments of the market. IBM also considered developing a radically new line of computers called Future Systems (FS) but various technical and organizational problems had led the IBM managers to cancel the project in 1975. In the same year, IBM entered the microcomputer market with a portable computer called 5100, which was targeted at a small community of scientific programmers. By late 1970s, the market for microcomputers had become the most rapidly growing sector of computer industry. Several companies had tried to capitalize on the growing interest of “computer hobbyists” in the production of microcomputers, of which Apple was the most successful. The main difference between microcomputers and other sectors of the industry was that microcomputers were targeted at individuals rather than corporate users. IBM, which after the failure of FS was looking for ways of maintaining its dominant position, decided to produce a microcomputer for the more common user, even though it was not a market that IBM was familiar with.

The team in charge of IBM PC made a series of key decisions that, within a few years, changed the shape of computer industry. The most important decision was to produce the IBM PC based on the components that were available in the market. This was in stark contrast to IBM’s general strategy (since System/360) that, unless under special circumstances, all components of IBM computers were produced within the company. The team also decided that the architecture of IBM PC was going to be open, meaning that IBM would make available (with the exception of one central component) the technical details of PC (again, against the traditional secrecy of IBM computer designs). Third, IBM PC was to be distributed mainly through independent sellers or retailer chains and IBM’s own Product Centers rather than the traditional sales offices of IBM.<sup>6</sup> Fourth, and in retrospective most significantly, IBM would not develop either application or system software for the PC and instead will rely on external sources to provide it. Some of these companies were

---

<sup>6</sup> For this purpose, IBM recruited and trained 700 dealers for the PC in the year before its announcement (McClellan, 1984: 216).

approached before the announcement of PC, but most were to be authorized later through IBM's 'software acquisitions' program.

After a study of the emerging microcomputer software companies of the time, IBM had chosen three companies to provide software for its PC: Digital Research Inc. (the leader in microcomputer operating system) for operating system, Microsoft for programming language and Peachtree Software for its software applications (primarily accounting applications). Digital Research and IBM failed to reach an agreement within the timeframe that IBM team had in mind and IBM asked Microsoft to provide the operating system. Microsoft, lacking experience, capability, or perhaps interest in developing an operating system from scratch, searched locally and decided to acquire the operating system that a neighboring microcomputer company (Seattle Computer Products) had created for its internal use. The company had no interest in commercializing its operating system (informally called Quick and Dirty Operating System) and for an undisclosed amount (no more than \$100,000) agreed to hand over the exclusive rights of the operating system to Microsoft. Microsoft also hired the programmer who had developed the OS. Subsequently, he and other Microsoft programmers enhanced and configured Q-DOS to match IBM's specifications and provided IBM with what became known as PC-DOS. Microsoft reserved the right to sell the same OS to other microcomputer manufacturers under the name MS-DOS.

IBM's decision to opt for an open architecture was partly driven by its urge for rapidly developing a new product and partly by the ongoing antitrust trial that had shaped its decision making for over a decade. IBM was conscious that an open architecture would enable other companies to reproduce its PC. In order to protect its PC from cloning by others, IBM had kept one essential component of PC as a black-box. The Basic Input Output System (BIOS) was piece of software that controlled all the signals between the operating system and the various hardware components. BIOS was copyrighted and IBM did not intend to sell it; in fact, it was embedded in a microchip on the mainboard of the PC.

IBM also envisioned a PC that could be run with more than one operating system. By creating a competition at the level of operating system, IBM was hopeful that no single company would have dominance over the PC operating system market and therefore IBM will not be over-reliant on a single company. Shortly after the agreement with Microsoft, IBM signed contracts with Digital Research and another company (called Softech) to provide alternative operating systems. Softech's operating system was available at the launch of IBM PC but Digital Research's arrived several months later and both were

overpriced compared to MS-DOS. MS-DOS became the most commonly used OS, and Microsoft became a key partner of IBM in the microcomputer sector.

IBM PC was introduced in 1981 and quickly became a spectacular success: the sale figures jumped from 25,000 in 1981 to more than 700,000 in 1983. From almost nothing, IBM had created a real threat for Apple, the most successful microcomputer company before IBM.<sup>7</sup> Though its contribution to IBM revenues was small (about 2% of company's revenues)<sup>8</sup>, for a brief period of time, the success of IBM PC established IBM as the market leader in the microcomputers and re-instated its dominant position in the industry. At the same time, it significantly raised the profile of the microcomputers and led to a flood of interest and investment in the sector. Soon, other companies started using the generous information published by IBM about the PC to build 'IBM-compatible' microcomputers that used essentially the same architecture as the PC.

IBM was hopeful that the copyrighted nature of BIOS would prevent other companies from copying PC and hence, despite the open architecture, IBM PC would remain an exclusive product that can only be sold by IBM. What IBM had not considered was that competitors could 'reverse engineer' BIOS and produce a software code that 'behaves' in the same way that BIOS does. The first company to succeed in doing so (after spending a reported \$1 million) was Compaq, which brought to market its cloned PC in 1982. By 1985, at least half of the microcomputer sector belonged to PCs of manufacturers other than IBM and IBM's share was decreasing. Microcomputers had become synonymous with PCs but the PC industry was increasingly independent of IBM.

As IBM's influence in the PC market declined, the influence of Microsoft increased. 90 percent of cloned computers were sold with MS-DOS installed and this brought Microsoft a steady source of revenue (though it was not charging much, less than \$50 when each computer was sold for more than \$1000). Moreover, building on its partnership with IBM, Microsoft became "the most opportunistic and diversified of the microcomputer software firms" with products ranging from add-on hardware cards to computer games (Campbell-Kelly, 2003: 207). Gradually, Microsoft turned away from relying on IBM and started to act as an independent software company with a significant and rapidly growing installed base of MS-DOS operating system. Although the two companies continued their partnership through

---

<sup>7</sup> Apple had virtually no profit in the year 1983, largely due to the competition from IBM PC and its cloners (see below).

<sup>8</sup> Figures are based on McClellan (1984: 216).

the 1980s, the relationship became increasingly thorny and both companies were looking for ways of breaking free.

At the same time, IBM PC and its clones gave a huge boost to the microcomputer software products industry. Up to that point, relatively few companies had been set up to provide software for microcomputers because computer hobbyists were not seen as a profitable niche market and there was no obvious role for the microcomputers in companies. The pioneering companies, Microsoft and Digital Research provided basic programming tools and operating systems. The success of Apple, Atari and other microcomputer start-ups had led to a small market for educational programs and games, but were not deemed appropriate for 'serious business'.

This perspective gradually changed as new start-ups began to provide so-called productivity tools, like spreadsheets, word-processor and database programs. (Well-known examples include Personal Software (1976, later Visicorp), MicroPro (1978) and Ashton-Tate (1980)). As scattered software companies sought to produce software for the buyers of the new PCs in their local markets, the 'personal computer' software industry took flight. This new industry, which was almost completely separate from the existing corporate software industry, soared on corporations' willingness to utilize PCs. Subsequently, a corporate market for the 'personal' computer software products was created (Campbell-Kelly, 2003: 2047). One survey in 1983 reported that one third of US companies were using microcomputers. According to this survey the percentage of large corporations (those with more than 5000 employees) using microcomputers was significantly higher than small companies (with less than 20 employees): 70% of large corporations -compared to 14.5% of small companies- used microcomputers.<sup>9</sup> Soon various kinds of business applications became stock-in-trade for the PC software companies.

Between 1981 and 1983, the sum of receipts of PC software companies had grown seven-fold but was still less than half a billion dollars. By 1983, there were nearly 3,000 software companies for the PC market offering a total of more than 35,000 different software products (Campbell-Kelly, 2003: 208-10). In contrast to the corporate software products, PC products were classic "consumer goods" or mass-market products: they required little or no training, mass marketing and exploitation of scale and network externalities. Whereas corporate software products (if successful) were sold in the range of \$5,000 to \$250,000 to

---

<sup>9</sup> *InfoWorld* (1983) 'Over one-third of U.S. businesses use micros' 19, Sep 1983.

(at most) a few hundred companies, PC software products were typically priced between \$50 and \$500 but were sold to tens of thousands of customers.

Most PC software companies consisted of a handful of programmers (some even one or two) and nothing more. Their most common marketing and sales strategy was using advertisements in magazines and mail-order for delivering products. They lacked manufacturing and retail capabilities that were needed for mass producing software (on diskettes) and offering them off-the-shelves. Soon, two important forms of intermediary emerged between software companies and consumers:<sup>10</sup> one, the software publisher, received the program from developers, duplicated it on diskettes, printed manuals, designed packaging and promotional material and produced a final product that could be offered to the customer in shrink-wrapped boxes. Some software publishers acquired the rights to certain (successful) products from their original developers and became software developing companies. The second intermediary, the retailer, offered software products through its retail stores and provided advice for customers. They also usually received feedback from their customers (especially unhappy ones) and reflected them to the software companies, making valuable (but often unrecognized) contributions to the subsequent development of software. The first outlets for PC software products were microcomputer retail chains but soon software-only retail stores started to emerge.

At the start of 1985, IBM seemed at least as powerful as it ever had been. The government had dropped the antitrust case against IBM in 1982 -claiming it to be “without merit”- even though IBM was still the dominant corporation in the computer industry (Delamarter, 1986: 25). In 1984, IBM had become the most profitable company in the US history, with after-tax profits of nearly \$7 billion (15% of total sales). Still, beneath the surface, the foundations of its domination were trembling. Flamm (1988: 240) observes that, since late 1970s, IBM had ‘strayed from the path of compatibility’ by introducing minicomputers that were neither compatible with each other nor with its mainframes. The introduction of PC had added to these problems. Moreover, the emergence of PC cloning companies (especially from southeast Asia) had made PC much less profitable than what it could be. But 1985 was to be remembered as the turning point in the history of IBM as well as the industry.

---

<sup>10</sup> A third group, essentially transportation companies, distributed products between retailers.

## 7.2. Analysis of the competition

While the structure of the industry was relatively intact throughout the 1970s, it was fundamentally transformed between 1981 and 1985. In 1969, mainframes were almost the only computers that were sold in the computer industry. By 1980, still 65% of computer sales in the industry went to mainframes. By 1984, microcomputers had grown to constitute 35% of the industry sales, while the share of mainframes had shrunk to 45 and minicomputers had modestly grown to 20%. Moreover, almost the entire growth in the sale of computers had come from microcomputers (and to a lesser degree) minicomputers (see Flamm, 1988: 238; Steinmueller, 1996: 27). Table 8-1 shows the value of US domestic computer consumption in the three sectors.

Year	Mainframes	Minicomputers	Microcomputers
1969	98%	2%	-
1975	90%	10%	-
1980	65%	20%	15%
1984	45%	20%	35%

Table 7-1: Share of each sector of the industry between 1969 and 1984

Though similar numbers cannot be accurately produced for the software and services industry (compare, for instance, Steinmueller, 1996: 28 and Cambell-Kelly, 2003: 18-9), by 1985, size of the industry had almost equalled that of the computer industry. A US Department of Commerce report estimated that, 3% of the industry (50-60 companies including IBM) had nearly 50% of the market. Commentators agree that even though the software and services industry (as a whole) grew rapidly (especially, in relation to computer sales) at the same time as PC arrived on the market, its growth was mainly the result of growth in the sales of software products which happened in the early 1980s. On the other hand, as we see below, the data processing which had flourished in the 1970s, suddenly began to decline in the early 1980s. Thus, most historians have associated the “hockey stick”-shaped growth of the industry with the arrival of “microcomputer revolution” and the subsequent growth in PC software products (see, for instance, Steinmueller, 1996, 20; Torrisi, 1998: 68; Campbell-Kelly, 2003: 13-17).

In both computer and software sales, IBM was the dominant company throughout this period. It maintained its dominant position in the mainframe sector and had a significant (though not dominant) share of PC market. Its software revenue was more than the software revenues of all other top 15 companies (including computer manufacturers and independent vendors). Apart from GE and RCA -who left the industry- the fortunes of other vendors from the mainframe era had ups and downs but was generally undramatic. Univac,



Honeywell and DEC, each became the no. 2 at one point or another. CDC, Burroughs and NCR grew consistently but were generally following the others. IBM's pioneering development of PC (compared to the BUNCH) and its success had put them once again in a reactionary position: within few years, UNIVAC, NCR introduced their own PCs. Burroughs, Honeywell and CDC never did.

Among IBM competitors, only CDC had significantly diversified its computer operations: by late 1979 mainframe production was the source of less than a third of CDC's revenues. By 1980, it had a marginally leading position with an about 7%-share of the market in data processing, which was nevertheless more than a third of its total revenues. Only three other companies -ADP, General Electric Information Services (GEISCO)<sup>11</sup>, and Tymshare Inc.- had market shares of 5% or more.<sup>12</sup> When the industry began to decline in the early 1980s many interpreted it as a result of the rise of PCs that provided data processing capability at lower costs. The industry reacted in various ways: CDC managed to continue its operations with the help of its other sources of revenue, GEISCO diversified by acquiring several software companies and ADP started selling packages of hardware and networking/application software to its customers (Fishman, 1981: 286-8, Campbell-Kelly and Garcia-Swartz, 2008).

### **7.3. The role of dissemination of skills in the transformation of microcomputers**

It is clear that the arrival of PCs (microcomputers in general), even in the few years that I have covered in this chapter, significantly affected the direction of industry. In this section I will focus on the role of seemingly unbounded dissemination of skills in the previous decades in the success of PCs. In my analysis, I will show that this dissemination was crucial for the rapid transformation of microcomputers from their hobbyist models to the mass-marketed PCs. Before doing so, however, I attend to a different issue that is equally important but its relationship with the transformation of PCs is less clear. As mentioned above, historians have found that independent software companies providing corporate and 'personal computer' software products were disjoint from each other. Since the structure of industry is important in shaping programmers work, it is important to understand the causes of this break and its role in the dynamics of software work. Therefore, I begin by trying to understand why corporate software companies (that were used to operating in several lines of activity) did not enter the PC software market. In my analysis, I will show that

---

<sup>11</sup> GEISCO was created when, in 1979, GE and Honeywell started a joint venture to provide data services using Honeywell-GE computers and GE software. The venture was successful and, in 1982, GE bought Honeywell's shares to make General GEISCO a wholly owned subsidiary of GE.

<sup>12</sup> Blumenthal, M (1980) 'Processing services now a whole new ball game', *Computerworld*, 1 Dec 1980.

individuals' skills raised no fundamental barrier for entry into the new market. What led to a barrier between the two sectors of industry was a mixture of issues rooted in experience and cultural differences. In exploring the nature of this gap, the role of dissemination of mainframe-era skills in the success of PCs becomes apparent.

The disjoint between PC software companies sector and the corporate software companies is surprising because, on the surface, their commonalities seem to exceed their differences. First, a significant bulk of them were purchased by corporations. Therefore, the two sectors had a significant number of customers in common. Second, before barriers to entry in each market were created, companies in both sectors were founded in a largely similar manner: a few programmers with a small initial capital and an idea that could be converted to a profitable product. Corporate software products could hire and assign a group of programmers to write software for PCs (as some did) Third, during its initial years, PC software market was structurally similar to the corporate PC market: competition in both industries had no clear leader but a small number of companies (together) had a disproportionately high share of the market. By 1985, even the revenues of top companies in both sectors were at similar levels. So there was no issue of imbalance in market power that could have prevented one sector from entering the other.

Therefore it was neither the customers nor entry requirements that separated the two industries. A few software companies that crossed the boundaries show the variety of possibilities that existed: Computer Associates (founded in 1972) entered both markets almost simultaneously (though it began with the mainframe market and acquired PC software companies later). Similarly, Management Science America acquired Peachtree in 1981. Cullinet, a successful provider of corporate software founded in 1968, started providing similar products for PCs in 1983. But most corporate software companies remained within their traditional market and boundaries between the two sectors remained in place. Perhaps most significantly, rather than joining the already existing Association of Data Processing Service Organizations (ADAPSO), which represented the software and services industry since 1972, PC software companies set up their own association (called Software Publishers Association) in 1984. Why did the two sectors remain separate? More specifically, why did not other corporate software companies follow Computer Associates, even after PC became a success?

Surely the technological differences played a role, but as Campbell-Kelly (1995) has argued, even when technologies had converged (by early 1990s) and applications had become functionally equivalent, the boundaries between the two sectors did not disappear.

Campbell-Kelly's own explanation is that cultural differences between those working in the two sectors are so different that crossing the boundary seems impossible. Other explanations emphasize the role of individual skills and firm capabilities. Cusumano (2004: 111), for instance, argues that the developing software for the new PC market required not only new technical skills but also different marketing skills. This observation is congruent with Campbell-Kelly's remarks about the different business models of the two sectors and the economies of products (capital goods vs. mass-market products, e.g. Campbell-Kelly, 2003: 8). More recently, Campbell-Kelly et al. (2013: 254) have argued that the tools and methodologies that existing software companies used for developing large reliable programs were "irrelevant" for the small memories of microcomputers.

I believe that these explanations are essentially correct but do not capture the complications that underlay the differences. At least, as far as founders are concerned, there seems to be a distinct generational gap between the founders of early PC software companies and those of the corporate mainframe era. Very few founders of new software companies had worked before the mid 1960s but most of them had short work experiences in large corporations in the late 1960s and 1970s (some who had stayed in universities had worked with companies on different projects). This common background shows that most of them had not only training in programming mainframe and minicomputers, but also work experience in corporate environments. Some of them had first started their programming jobs while they were still high school or university students, using the flexible labor market environment of the 1960s. The few individuals who were significantly older (born in the 30s) either concentrated on marketing functions (e.g. Lashlee, Rubenstein) or were pushed out (e.g. Davis, Canova).

Nevertheless, the fact that some seasoned mainframe programmers could emerge in marketing/publishing roles shows that the experience (and connections) that were built in the corporate environment could be fruitfully transferred to the PC software market. Similarly, the gap in technical skills was mostly a matter of conventions and protocols rather than basic programming skills (see Frankston's comments in Bricklin and Frankston, 2004: 26). It appears that the generational gap between the founders of early corporate product companies and those of PC product companies had given rise to a mixture of experience and cultural differences. The first group were a species of "organization man" (Whyte, 1956) who had witnessed developments of computer technology inside corporations and were used to it. In contrast, the latter group were relatively new incomers who had different perceptions about both the work environments and the new technologies.

<b>Name</b>	<b>Work Experience</b>	<b>Company Founded (Year)</b>
Paul Allen (1953)	TRW, Honeywell (1972-73)	Microsoft (1975)
Alan Ashton (1942)	Brigham Young University, BYU (1973-87)	Wordperfect (1978)
Bruce Bastian (1948)	BYU (1973-7), Eyring Research Institute (1978)	Wordperfect (1978)
Dan Bricklin (1951)	MIT, DEC (1973-76)	Software Arts (1979)
<i>George Canova</i> (1931)	RCA, Burroughs, Scientific Data Systems (?-1968)	Novell (1981/3) <sup>13</sup>
Doug Carlston (1947)	Iowa City University (1965)	Broderbund (1980)
David Crane (1953)	National Semiconductor (1975-77) <sup>14</sup>	Activision (1979)
Jack Davis (1939?)	NCR, GE, Scientific Data Systems (1961-1970?)	Novell (1981/3)
Ben Dyer (1948)	AT&T (1970-2), Computer System Center (1977-8)	Peachtree (1978)
Richard Frank (1943)	CDC (1965?-79)	Sorcim (1980)
<i>Bob Frankston</i> (1949)	Various programming jobs (1966-8) Interactive Data Corp. (1969?-79)	Software Arts (1979)
Dan Fylstra (1955)	San Diego State University (1972-3) Intermetrics (1975-6)	Personal Software (1978)
Bill Gates (1955)	TRW, Honeywell <sup>15</sup> (1972)	Microsoft (1975)
<i>Fred Gibbons</i> (1949)	University of Michigan, Ann Arbor (1970-2), HP (1975-81)	Software Publishing (1980)
<i>Martin Herbach</i> (1943?)	CDC (1965-1980)	Sorcim (1980) <sup>16</sup>
<i>Steve Jasik</i> (?)	CDC (1967-1984)	Sorcim (1980)
Philippe Kahn (1952)	HP (1982) <sup>17</sup>	Borland (1982)
Larry Kaplan (1950?)	Control Systems Industries (1974-6)	Activision (1979)
Mitch Kapor (1950)	An unnamed computer consulting company (1973) <sup>18</sup>	Lotus (1982)

<sup>13</sup> Novell's founders (Davis and Canova) had been involved in the manufacturing firm California Computer Products Inc. (CalComp). Novell too started as a plug-compatible manufacturer but the founders left within a year and the company was turned into a software company by 1983.

<sup>14</sup> Crane worked in hardware design.

<sup>15</sup> He was offered the job at the same time as Allen. Accounts vary as to whether he worked only during the summer or did not work at all.

<sup>16</sup> Sorcim's other co-founders (Anil Lakhwara and Steve Jasik) were all CDC employees but little biographical information is available about them.

<sup>17</sup> Kahn was fired after three days, because HP discovered that he was in US on tourist visa.

<sup>18</sup> Kapor worked as marketing analyst.

<b>Name</b>	<b>Work Experience</b>	<b>Company Founded (Year)</b>
Gary Kildall (1942)	Intel, Naval Postgraduate School (1972-3)	Digital Research (1974)
<i>Hal Lashlee</i> (1941)	A mortgage bank (?-1979?) <sup>19</sup>	Ashton-Tate (1980)
<i>Paul McQuesten</i> (1948?)	CDC (1970-8)	Sorcim (1980)
Alan Miller (1951?)	Various Silicon Valley companies (1973-7)	Activision (1979)
Vern Raburn (1950)	3M (1976)	Microcomputer Software Associates (1977)
Seymour Rubenstein (1934)	A defense contractor (1963?-1970)	MicroPro (1978)
Jonathan Sachs (1947)	Several research organizations (1966-70) MIT (1970-6) Data General (1977-1979)	Lotus (1982)
George Tate (1944)	Several computer companies (1975-9)	Ashton-Tate (1980)
Dennis Van Dussen (1948?)	First Data (1976)	Personal Software (1978)
Bob Whitehead (?)	A defense contractor (?-1977)	Activision (1979)
<i>Ken Williams</i> (1954) <sup>20</sup>	Various LA companies (1973-79)	Sierra Online (1980)

Table 7-2: Names and work experience of pioneering PC entrepreneurs. Names *in italics* had more than 5 years of experience in the corporations.

These differences suggest, tentatively, that these individuals were better suited to question the predominant perspectives of industry. This could have contributed to their appreciation of potentials of microcomputers when they appeared in their early forms. It is well known that the mainstream computer industry was largely dismissive of the potentials of the new microprocessors, or as they knew it “controllers”.<sup>21</sup> By being able to break with the dominant perspectives of the industry and, at the same time, build on the skills they had gained while training or working in those environments, PC software entrepreneurs could seize the opportunity to create a new sector. It was this distinction that also set them apart from those working in the corporate software industry. As we will see these differences among computer workers became entrenched over time to a degree that sometimes resulted in bitter arguments (as in the case of discussions about the future of COBOL in the early 1990s, see next chapter).

<sup>19</sup> Lashlee was an accountant by training.

<sup>20</sup> Ken was a graduate of Control Data Institute.

<sup>21</sup> Rogers and Larsen (1984: 106-8) document Intel’s own doubts about announcing the new chip. Gary Kildall set up his own company to sell the CP/M when Intel refused to buy it, apparently because they could not see any use for it.

This distinction also puts early microcomputer software entrepreneurs on a par with the computer hobbyists, who are often understood as mediators that helped the metamorphosis of microcomputer from its early forms to the user-friendly screen and keyboard-based tool that ‘almost everybody’ could learn and use (see Campbell-Kelly, et al. 2013: 229-45). Campbell-Kelly, among others, has noted that the personal computer “had a surprisingly long gestation” in which computer hobbyists were the only customers and ended with mass-market production of personal computers in the 1980s. While exploring this process of diffusion of technology and “democratization” of skills is beyond the scope of this research, a brief discussion of the role of these groups can shed further light on the role of widespread availability of skills in the transformation of microcomputers from their early forms to the PC era.

One can consider the position of computer hobbyists vis-a-vis microcomputer software entrepreneurs from two aspects. On the one hand, computer hobbyists generally belonged to the same generation as those entrepreneurs. In the words of Campbell-Kelly and his colleagues, the computer hobbyists were “typically young male technophile[s]”, “often employed as technicians or engineers in the electronics industry” whose “enthusiasm for computing had often been produced by the hands-on experience of using a minicomputer at work or in college.” On the other hand, they had a point of sharp difference with the software company founders: computer hobbyists were “primarily interested in tinkering with computer hardware; software and applications were very much secondary issues.” This particular combination had made Altair (and other early microcomputers) an “unprecedented” and almost irrational design: “it would appeal only to an electronics hobbyist of the most dedicated kind, and even that was not guaranteed” (p. 232-3). Very few in the (hardware or software) industry could make sense of its existence. To many, it was at best a curiosity gadget.

This put microcomputer software entrepreneurs in a very peculiar position: on the one hand, they like many others, had skills that could not be directly used on the microcomputer. Even though they had learned their programming skills by working on mainframes or minicomputers and had used tools such as programming languages, it appeared that they had to do everything from the scratch in order to make the microcomputers work. As Campbell-Kelly, et al. (2013: 236) put it:

“[P]rogramming the first personal computers had many similarities to programming a 1950s mainframe: there were no advanced software tools, and programs had to be

handcrafted in the machine's own binary codes so that every byte of the tiny memory could be used to its best advantage.”

On the other hand, they could see the opportunities for the new computer, if it was developed in ‘the right way’, which, as they understood, was a mass-product “personal” tool. But in effect they had to appeal to those individuals that had skills to further utilize it. In other words, their success depended on relating this opportunity to the available pool of skills. Indeed, this is what Gates and Allen did when they developed their own version of BASIC for Altair: they tapped into the vast pool of enthusiasts who had BASIC skills.<sup>22</sup> Subsequently, other programmer/entrepreneurs used these and other tools to develop software, most famously for Apple II but also for other microcomputers (Evans, et al., 2006:86-7). Over time, improvements in microprocessors as well as applications (which was a result of interactions between software developers and hobbyists) led to development of products that could be used more easily and by more people.

At some point during these interactions the bonds that held hobbyists and software developers together broke. This is often marked by the success of VisiCalc, which -at least in the folk history of software is credited as a software product “that transformed the perception of the personal computer as a business machine” (Campbell-Kelly, 2003: 203). This is despite the fact that, at the time of its release, it was ignored by the mainstream media and dismissed by both the mainframe industry and a majority of hobbyists. According to Bricklin (in Bricklin and Frankston, 2004: 24), the latter group rejected it because it was “business oriented” rather than “hobbyist oriented”. But even then, it took sometime for the software companies to provide successful products for the corporate environment.

Thus, the new generation of software entrepreneurs managed to build on their mainframe-era programming skills to develop successful business models. The contrast is perhaps most visible in IBM, which had put in place a software acquisition program for commercializing programs developed by its employees. Despite the fact that many IBM employees were first to buy IBM PC, not one piece of software was acquired internally.<sup>23</sup> In all likelihood, those with some idea were more ready to pursue them outside rather than try to convince their managers that their idea was good. We know, for example that the sources of many early games were “moonlighting programmers”, otherwise known as ‘authors’. (Campbell-Kelly,

---

<sup>22</sup> Gates and Allen effectively developed an “interpreter” for BASIC programming language on microprocessors. This can be contrasted with Kildall’s new programming language “PL/M”. Other developers had developed simplified versions of BASIC for microprocessors independently at about the same time as Gates and Allen.

<sup>23</sup> Initially, IBM only developed programming tools (like emulators) for its PC, and later some software products. But these were through its normal operations rather than employee initiatives. See Shea, T (1983) ‘IBM content to let others supply software’ *InfoWorld* 28 Dec 1983.

2003: 210). Nevertheless, there are few former-IBM programmers in the list of early software entrepreneurs. Thus, while the failure on the part of mainstream computer industry to appreciate what microprocessors could do was reverberated in the failure of established software industry in understanding what uses it could have, software entrepreneurs used the flexibility of labor market and widespread dissemination of skills to create a new and distinct industry. In time, this new software industry would not only change the structure of the market, but also shape a new business model for software and a new employment relationship for software work.

#### **7.4. Varieties of software work**

The generational gap that I highlighted between the founders of corporate software companies and the sample of personal computer software entrepreneurs does not imply that there was a generational gap between those who worked in the two sectors. Nor does it mean that all the new entrants in the labor market sought to be employed in the ‘newer’ industries or that “the mainframe people” were a dying creed. Nevertheless, as I will show in this section, divisions in the structure of industry in this period had consequences for institutionalized forms of employment relationships as well as formation and distribution of skills. In other words, while companies in each sector did have options, the decisions they made not only changed the emphasis on what was important in the employment relationship, but also gave rise to new justifications and rationalization about those choices. By the end of this period, the old (but not necessarily observed) ‘norms’ of employment relationship were being gradually replaced by a new arrangement.

I begin by discussing the computer industry and the challenges companies faced (individually and as an industry) during the 1969-1985 period. I show how these challenges affected both companies short-term employment decisions and long-term policies. While my discussions are not exclusively about software workers, in all cases software workers were affected. In the case of each company I focus on specific localities that demonstrate the changes in employment relationships. Then I discuss the software industry, especially practices in the emerging PC software sector. I will limit the main examples of my discussion to California because it had both the highest concentration of software companies/workers<sup>24</sup> and was considered as the hotbed of new employment practices. I will conclude by showing that how these new employment practices gave rise to new forms of

---

<sup>24</sup> It was estimated that roughly 20% of nation-wide jobs in the software industry (about 250,000 jobs) were in California (Markusen, 1983). At least a third of those jobs in programming and systems analysis, 44% keypunch and computer operators, sales personnel and other clerical staff and finally, 14% managerial jobs (Hall, et. al. 1985).



skill formation. This will lead to discussion about the changes in the labor market in the next section.

#### **7.4.1. The computer industry**

As I mentioned in the previous chapter, during the 1950s and 60s, the majority of user organizations employed programmers and developed their own systems. Nevertheless, this often happened with the help of vendors' programmers, who in many cases had on-site presence throughout the stages of development of those systems. Therefore, most of the software work (including these support activities) were done by employees of computer vendors and the independent software and services industry had a small share of software workers. In the previous chapter I also outlined the employment policies of some of these companies and highlighted the particular importance that was attached to employment security.<sup>25</sup> I also highlighted that the continuous expansion of computer market throughout the decade was reflected in continuous increase in the number of workers and therefore, promises of employment security were never really tested. The recession of the beginning of the 1970s tested the employment relationship between employers and employees in the computer industry for the first time. Another economic recession happened following the oil price shocks of the late 1970s but did not seem to affect the computer industry as a whole.<sup>26</sup> Nevertheless, some companies changed their employment policies in the aftermath of these developments. Given the prominence of IBM in the industry and its 'progressive' employment relationships, I will first consider the decisions of IBM and then those of other computer companies in relation to their employees during the period between 1970 and 1985.

#### **IBM**

IBM had grown rapidly in the years following the successful launch of System/360. From around 150,000 employees in 1964, it had grown to 258,000 in five years (until 1969).<sup>27</sup> This rapid increase hit a peak of 269,000 in 1970 and then was decreased to 262,000 in 1972.<sup>28</sup> According to Fishman (1981, 90) IBM 'clung' to its no lay-off policy but nevertheless reduced the number of its personnel in Data Processing Group (responsible for computer marketing, development, manufacturing, and related services) by 12,000 workers over two years. IBM managed to accomplish the cuts through attrition and freezing the

---

<sup>25</sup> Employment security, as I will discuss below, is less restrictive from job security: the former concerns continuous employment opportunities for the worker while the latter guarantees that the worker can remain in the same job.

<sup>26</sup> Nearly 10.8 million Americans lost their jobs in the early 1980s (Mills, 1988: 8).

<sup>27</sup> In comparison, its growth from the start of the decade till 1964 was less than 50,000.

<sup>28</sup> It must be remembered that these numbers are annual aggregates and do not reflect turnover or simultaneous hiring and layoffs.

hiring. It also provided an early retirement option. It is unclear to what extent software workers were affected during this period but it appears that most of the ‘surplus’ was deemed to be in the overhead staff and manufacturing departments. Many of people in those jobs were subsequently moved to sales jobs.

Year	1964	1965	1966	1967	1968	1969	1970	1971	1972
Employees (thousands)	150	172	198	221	242	258	269	265	262

Table 7-3: Worldwide number of IBM employees (in thousands).

One reason that the effects of these decisions on software workers cannot be traced is that IBM never (until 1992) had a centralized software group. Thus -apart from those who provided field support to customers throughout the country- IBM software development groups were scattered throughout the country. As early as 1962, one programmer ad listed the following locations for programming facilities: New York City, Endicott, Kingston, Owego, Poughkeepsie and Yorktown Heights (NY), Lexington (KY), Rochester (MN), Omaha (NE), St Jose (CA) and Washington D.C. area. The plants or offices in each of these locations had software workers (including programmers) working on various projects that belonged to different lines and products. In 1968, it was estimated that IBM had 7,000 employees working on programming (Humphrey, 2002).<sup>29</sup>

Nevertheless, it is possible to get a glimpse of the work and life of these workers by focusing on the local newspapers of the time. I mentioned in the previous chapter that in many locations in which IBM had an office or plant it became the major employer in the area. This greatly affected the shaping of local communities and their lives. Moreover employees that stayed with the company for decades were likely to have another member of the family (like their children) who also worked in IBM. For example, the local newspapers of the Hudson Valley and Endicott (where the New York state plants and offices are located) are filled with the news of engagements, weddings, promotions and retirements of IBM employees. The engagement or wedding items often mention the education and current position of IBM employees while promotion and retirement items reflect the positions that an employee had and the achievements he or she had made.

---

<sup>29</sup> This number is produced by modifying what appears to be an error on the part of Humphrey. Humphrey notes that, in 1968, “IBM had about 130,000 employees, about 4,000 (or 3 percent) of whom worked on programming.” He is recounting the basis of his argument, made in 1968, that IBM’s reduction in hardware price following the unbundling reflects the percentage of employees that work on software. But in 1968, IBM had 242,000 workers. It is possible that Humphrey has made a mistake in remembering the accurate numbers. If so, for the argument to hold, the number of software workers must have been around 7000.

As far as software workers are concerned, the information included in these items confirm the general observation that entry to software jobs (programmers, operators, system engineers, etc.) did not require any specific qualification. More importantly, the promotion and retirement pieces reveal another aspect of work in IBM: the absence of rigid career paths and flexibility in the allocation of individuals to jobs even when it implied lateral movement. It was not unusual for an employee to start as a mechanical engineer on the production line and end up as a programmer in a software group within a few years. This was, at least in part, out of necessity: it was impossible for IBM to commit to its no lay-off policy without such flexibility (see Loseby, 1992:118-123; Mills, 1988: Ch. 6). In other words, if the company wanted to enforce occupational or functional barriers to movement of people between jobs, it had to lay off workers in one department and hire new workers in another. Instead, IBM's strategy was based on retaining and retraining workers. For instance, between 1970 and 1975, IBM "retrained and physically relocated 5,000 employees as part of the most extensive corporate education program in the U.S."<sup>30</sup> Another 5000 were retrained to take up new jobs in 1984 alone. Walton Burdick, IBM's personnel vice president told a Business Week reporter that this had happened because "skills and resource needs significantly changed" (Loseby, 1992:118-121).

In case of computer-related jobs, with their poorly defined entry requirements and seemingly endless demand, this meant that there was a continuous flow of people from other occupational backgrounds to these positions which were almost always better paid than employee's previous jobs. (Those who left programming, on the other hand, often went to jobs that were more respected within the IBM like management or sales.) The case of one employee in Boulder (CO) illustrates this point (Mills, 1988: 115-120).

Bob Bailey was a former Air Traffic Controller in his mid thirties and without a job. He, like many of his co-workers, had lost his job following participating in the strike by PATCO (Professional Air Traffic Controllers' Organization) members in 1981.<sup>31</sup> After taking some odd jobs (in petrol stations and elsewhere), he followed the advice of his neighbors who worked for IBM and applied for a job at IBM in 1982. Having no skills that were of use to the Boulder manufacturing plant, he had applied for a job as an assembler or a shipping clerk. After interviews, IBM hired him as a temporary forklift driver and in the spring of 1983 IBM offered him a full-time job as a forklift driver. However, in his new job he earned about a third of his previous income and looked for ways of advancing his position. In mid 1984, he became aware of an opening in the computer room of the plant for a computer

---

<sup>30</sup> Business Week (1975) 'How IBM avoids layoffs through retraining', 10 Nov. 1975.

<sup>31</sup> The strike was crushed by the newly elected president, Ronald Reagan.

operator. He talked to the manager and after a few months was transferred to the computer room.

In late 1985, company made its employees aware that there was a serious shortage of systems programmers and invited all interested employees to apply. He and nearly one thousand other employees in Boulder applied. After seating at aptitude tests, a hundred employees were invited for interviews. In the end, Bob was one of the forty people who were selected to be trained to become a programmer. The programming course began in Feb. 1986 and they were trained 40 hours per week for 15 weeks. The training course was followed by on-the-job trainings, which lasted several months, coinciding with the transfer of the Westlake programming unit. By 1987, Bob had received his “associate programmer” certificate and a substantial advance in level and payment. This was by no means an exceptional case. His cohorts included a psychology graduate and former plant worker-turned-manager who had taken a cut to become a programmer and a former senior design specialist.

It is important to note that, as Bob’s early experience with IBM shows, IBM achieved flexibility by keeping the headcount lean and relying on contingent contractual workers as buffers. IBM departments were consistently under-staffed, at about 85% of the level that was actually required. In normal periods, the extra 15% was provided for by overtime work of employees and, when necessary, a small number of temporary workers. When demand was high, contingent workers were brought in large numbers. Thus, IBM was able to cope with demands levels between 85% and 115% of the normal level (Gutchess, 1985: 41). While these strategies are usually discussed in relation to blue-collar work, they were also useful in the context of software work, with its unpredictable workload and often-missed deadlines. To put the numbers in context, the norm of understaffing in the industry at the time was around 70% with some companies going as low as 50% (p. 40). Moreover, given the close relationships between IBM managers and employees in their communities, IBM managers were generally aware and considerate of the conditions of the temporary workers.

IBM managers were proud of their record and used to boast about it (see Cappelli, 1999: 251n.1). After the antitrust trial was dropped in 1982, IBM President John Opel used to say “the only monopoly that IBM has is in the way it treats its people”. They knew that other companies could not combine the same level of flexibility and profitability. As we will see below, IBM had the special position of being considered an exception and, at the same time, setting the standard for the industry; it was the example that others tried to follow, often unsuccessfully.

\*\*\*\*\*

IBM was not alone in its commitment to employment security. HP was another well-known employer who provided such security while Data General and DEC only did so until early 1980s, when they decided to abandon it to cut costs. Other companies that could not provide employment security took steps to stabilize their employment relationships. Below I discuss two examples that illustrate the challenges that companies faced and the options managers had (see also Kochan et al, 1988, for a discussion of DEC).

### Honeywell

Following the acquisition of GE's computer division, Honeywell's headcount jumped nearly 20,000 to above 100,000.<sup>32</sup> It now had around 25,000 employees in its computer division (Honeywell Information Systems) in the U.S. (and a similar number overseas), of whom about 75% worked in white-collar jobs (including managers, technical workers and sales and service personnel). Throughout the early 70s, Honeywell was forced to trim its number of employees. For a few years afterwards Honeywell expanded but in terms of number of employees remained below the 1970 peak.

Software work was one of the major areas of expansion of Honeywell in the late 1970s. Ever since the 1960s, Honeywell's main software development operations were based in Phoenix (AZ) and Wellesley (MA). But like other manufacturers its programmers were widely scattered around the country. The annual reports of a newly set up unit in Los Angeles can illustrate some of the challenges that the company faced in managing its employment relationships in relation to software workers. Following acquiring a part of Xerox computer operations in 1976, a new unit called Los Angeles Development Center was set up to provide support for Xerox users. Within a year of its establishment the number of employees grew from 66 to 119. Nearly all of these employees were software workers. A significant part of the growth came from rehiring former Xerox employees. Newspaper and college ads as well as employment agency and employee-referrals were also used. Nevertheless, it was felt that there was need for even more workers. Furthermore, recruitment efforts were complicated by voluntary attrition of 18 people in the same year. In the subsequent year the headcount had risen to 147 (including more than a dozen contractual workers) but 1 in 5 of employees had left since previous year. Unit's annual report cited "the pressures of the very aggressive

---

<sup>32</sup> Long-time GE and Honeywell programmer, Richard Coupe, writes in his biography that 3200 GE employees were "relocated, laid off, or hired by several other firms GE actually invited to Phoenix to conduct job interviews." (Coupe, 2013: 96).

recruiting marketplace for systems programmers in Los Angeles” as well as budget limits and organizational delays as sources of the problem in attracting more employees.

Nevertheless, during the economic downturn of the 1981-2 period, Honeywell laid off nearly 3000 people, many of them in the computer division. Honeywell managers were explicit in their recognition of “current and future shortage of highly qualified technical personnel” and “competition from other companies providing employment security guarantees” (Gutchess, 1985: 58). The company was also aware of the limits of “traditional layoff and recall” process and therefore charged its Organizational Development Advisory Board “to define, evaluate and recommend policies for improving employment stability in Honeywell” (quoted in Gutchess, 1985: 55). While providing lifetime guarantee of employment was not an option, the company believed that organizational flexibility can bring stability for the workers and growth for the company simultaneously. One key element of these policies, as in IBM and others, was providing continuous training for employees “particularly in relation to the constant need to keep abreast of changing technology in the fields in which the company operates” (Gutchess, 1985: 58).

\*\*\*\*\*

#### CDC

Between 1957 and 1969, CDC had grown from a company of 12 to 45000 employees. During the mid 1970s it had laid off some 6% of its employees. But in the following years grew and reached 60,000 employees in 1981. Between 1979-1980 its growth had stalled but the company had resisted massive layoffs. William Norris told his executives at the time that:

“In this day and age,...immediate layoffs in response to a decline in business are becoming increasingly unacceptable in our society. Nor is it acceptable to abruptly close plants as soon as there is lack of need, without reviewing the reasons and possible alternatives with the affected community.”

A Job Security Task Force was initiated in 1980 to devise policies that would help provide employment security. It stated that CDC’s objective in its employment relationships was to provide “an increasing level of job security to the greatest number of its employees”. Various practices were considered and ultimately a “ring of defense” strategy was devised to stabilize employment levels. This included understaffing (at 70% level), use of part-time workers (on a permanent basis) and personnel of small entrepreneurial businesses that CDC had supported, and finally contracting part-time workers. The idea behind using temporary part-time workers, as the company’s spokesperson had explained, was to rely on the pool of individuals who are “specifically looking for a non-permanent supplement to their income”

(see Gutchess, 1985: 46-7). The company also devised a strategy to prioritize its action with regards to each group of employees, exploring alternatives such as hiring freeze, work hours reduction, voluntary layoff, performance-based layoffs, retraining and redeployment, etc. Only when all these options were tried and failed, the company would lay off people involuntarily.

But as we saw, the CDC's real troubles started in the early 1980s. In retrospect, its policies seem to have been utterly ineffective. Between 1981 and 1985, the number of employees in its various computer operations had fallen from more than 52,000 to 39,500. Number of CDC employees in Minnesota alone fell from above 25,000 to 15,000. It did not however stabilize at those levels and it continued to slide afterwards. While the direct implications of these layoffs for CDC software workers is not clear, it is certain that they were affected by the overall decline in the viability of the company. Whether or not company intended to maintain them, they probably started to look for opportunities outside and "jump the ship" before they were forced to. We have already seen that some of the founders and early employees of Sorcim software company left CDC in this period.

\*\*\*\*\*

Many other companies that struggled in this period resorted to layoffs, including Burroughs, Sperry, Apple, Atari, etc. In Atari alone, 80% of employees (4800 people) lost their jobs between 1983 and 1984 including many software workers.<sup>33</sup> Even companies that had massive layoffs (like CDC, Wang, etc.) offered counseling and job-placement services for their former employees, providing them with training, advice and assistance to find a new job. The most important motive behind these efforts was that many in the industry felt that employment security was "increasingly important" (Gutchess, 1985: 54). They wanted to maintain their relationships with their employees and in some cases re-employ them at a later stage.<sup>34</sup>

As an exception that proves the rule (and also indicates the changing attitudes), one can consider the terms that Apple offered to its employees. Beginning in the 1980s, the following statement appeared in the employment contract of every Apple full-time employee:

"Here is the deal we will give you; here is what we want from you. We're going to give you a really neat trip while you're here. We're going to teach you stuff you

---

<sup>33</sup> *InfoWorld* (1985) Atari layoffs continue as product line expands, *InfoWorld* 24 Jun 1985  
*InfoWorld* (1984) Atari: From starting block to auction block, 6 Aug 1984

<sup>34</sup> Burton, K (1985) Beating layoffs a matter of skills, *Computerworld* 8 Jul 1985

couldn't learn anywhere else. In return, we expect you to work like hell, buy the vision as long as you're here.... We're not interested in employing you for a lifetime, but that's not the way we are thinking about this. It's a good opportunity for both of us that is probably finite." (Quoted in Ettore, 1994: 15; see also Cappelli, 1999: 25)

The approach of Apple managers to the employment relationship could hardly be more out of line with the prevailing views of the industry<sup>35</sup>. But the software industry, to which I now turn, had dealt with its employees in the same way for a long time, probably without having written down anything.

#### ***7.4.2. Software companies***

I have mentioned that the number of companies in the software and services industry grew from the mid 1960s onwards but a large number of these companies were too small and too short-lived to leave a record. As one may expect, these companies emerged in clusters near their potential customers. Thus, on the west coast they were found primarily in California and especially in Los Angeles and later, San Francisco. Companies on the east coast were more evenly distributed between Boston, New York, Philadelphia and Washington D.C. area. Other cities that hosted a significant number of companies were St. Paul (MN), Chicago (IL) and Houston (TX). By 1974, the top 10 metropolitan areas accommodated 70% of all employees in the software industry (Egan, 1997: 54).

For much of the 1970s, the dynamics of employment relationships in most software companies was similar to 1960s: companies relied on a handful of programmers and had to strike a balance between keeping highly-paid programmers and having enough customers to make a profit. The trend towards diversification had helped some companies to stabilize and the most successful companies had hundreds of employees. Nevertheless, in an environment dominated by IBM, offering employment security (even if possible) was not especially attractive. Software companies, therefore, emphasized that beyond benefits and stock options they provided "a challenging environment", "opportunities for growth", and of course, "flexible" salaries.

As mentioned before, many corporate software companies had a diversified business model that comprised data processing and programming services as well as software products. This necessitated not only a variety of technically skilled workforce (in systems analysis and

---

<sup>35</sup> It is notable that until late 1970s, Apple did not consider software an important part of their business model. For a description of position of programmers in early years of Apple, see Mortiz (1984).



programming as well as marketing and sales), but also a geographical distribution of employees. When the new software companies for personal computers emerged, their exclusive focus on software products meant that in their business model, software work was concentrated in the main software development process and, to a lesser extent, (telephone-based) customer support. Since those products were aimed at mass-markets, third parties and sales personnel who no longer needed to be technically sophisticated often were responsible for retail sale. At the same time, having access to a pool of programmers was extremely important. While many decided to be close to areas where programmers could be easily found (like New York, Los Angeles and San Francisco) some decided that keeping their employees away from the competition was more important. In December 1981, just before they were shot into global fame, Microsoft's Paul Allen gave this explanation to *ComputerWorld's* reporter about their decision to set up their base in Seattle:

“Silicon Valley was one of the two places we could go, but there is a very high turnover in employees there; everybody is raiding everybody else. The other choice was Seattle, which has a relatively good base of programmers and it's not difficult to get people to move here”.<sup>36</sup>

In mid-1980s the rapid growth of PC software companies came to a momentary halt and the first waves of layoffs followed very soon. For example, MicroPro, which had grown from a 2 person company to more than a thousand employees in four years, laid off near 600 people between 1982 and 1984. Commenting on the layoffs, the company announced that those laid off were given “no promises about rehire”.<sup>37</sup> Similarly, Digital Research, Software Arts, Visicorp, Activision and many other companies were shaken during the ‘slump years’ and many others did not survive.<sup>38</sup> The corporate software companies seemed to be relatively less affected by the market fluctuations, even though some companies reduced their employees without making any announcement (like MSA, which “quietly” trimmed near 5% of its 1900 employees).<sup>39</sup> Such layoffs occurred in many metropolitan areas like around Boston, New York and other cities.<sup>40</sup>

---

<sup>36</sup> Batt, R (1981) Small software house finds room to grow, 7 Dec. 1981.

<sup>37</sup> Caruso, D (1984) Micropro lays off 62 workers, *InfoWorld* 28 May 1984

<sup>38</sup> Gallant, J. (1985) Causalities mounting in heat of micro market battle, *Computerworld* 30 Jul 1985. Gallant, J. (1985) Slump moves to software, *Computerworld* 22 Jul 1985. Gallant, J. (1985) Software vendors now feel pinch, *Computerworld* 15 Jul 1985.

<sup>39</sup> *Computerworld* (1985) Top of the news, *Computerworld* 25 Nov 1985.

<sup>40</sup> Howitt, D (1984) Knowware feels the pinch, *InfoWorld*, 29 Oct 1984. The much celebrated features that are supposed to distinguish west coast (‘Silicon Valley’, representing network organizations) and east coast (‘Route 128’, representing hierarchical corporations) are much more stylized than accurate and overlook broader institutional arrangements and tendencies (See Hyde, 2003; 258 cf. Fligstein, 2008).

Financial difficulties had always been ‘acceptable’ justifications for layoffs, even though whether or not they applied to any specific case was disputable. What is interesting in this context is that some software companies actively played down these issues in favor of other justifications. Commenting on these alternative explanations, Computerworld editors wrote “[...] we’ve heard some interesting rationales for the layoffs - ranging from the need to get “lean and mean” to “we’re back to being an entrepreneurial company”<sup>41</sup>”. Most of these explanations referred to organizational restructuring efforts which were becoming increasingly common. But perhaps the most unusual justification was provided by the CEO of MicroPro who after laying off 62 employees released a statement in which he claimed that the layoffs were part of a “regular review by management of activities within the company”. He also claimed that the company was “observing some new trends in the market” and the shift would allow the company to develop new products and reach the market more quickly”.<sup>42</sup> In other words, these layoffs were not related to financial difficulties, organizational restructuring or employees’ underperformance; the company was terminating those employees because they were not deemed to be valuable in the new market.

### **7.4.3. Analysis**

The changing employment relationships are important because of their implications for both the competition and the dynamics of skill formation. Commenting on the importance of turnover rates in the dynamics of competition in the US computer industry, Flamm (1988: 218) wrote in a footnote:

“IBM comes to mind as a firm widely known for excellent employee compensation and employment security, as well as lower turnover. Rather than reflecting an uncommon measure of benevolence, this may well be a highly rational strategy in a business preserving the security of internal technical know-how”.

IBM indeed was very aware of the risks of losing valuable workers. Especially with the rise of plug-compatible manufacturers, many inside the IBM saw more opportunity outside rather than inside and began what Pugh (1991: 490) calls “defection en masse”. In response to this IBM began a review process. They found out, for example, that sales and marketing people had turnover rate of 7%, twice or thrice that of few years before (but still very low compared to industry standards).<sup>43</sup> In order to bring the defection rate under control, they

---

<sup>41</sup> Needle, D (1984) From the news desk, *InfoWorld* 30 Jul 1984

<sup>42</sup> Caruso, D (1984) Micropro lays off 62 workers, *InfoWorld* 28 May 1984

<sup>43</sup> In 1990, a survey of 101 former system engineers and sales personnel who had left IBM before 1980 reported that most of them were in similar jobs in other companies. Others either were working in data processing departments or had founded their own (software and services) companies.

instituted an exit interview procedure in which three levels of management interviewed those who wanted to separate and tried to persuade them not to do so (Fishman, 1981: 88-9).

Still, whether or not IBM's employment policy was used as a "highly rational strategy" is an open question. The rationality of actors must be understood in the institutional context that they are operating, and therefore any analysis that focuses solely on economic rationality misses the point. In the case of IBM, we saw that part of the urge towards 'progressive' employment practices was to avoid unionization and instability. But after their peak in the 1950s (the McCarthy era) unions had increasingly lost their footing in American businesses. At the same time, those 'progressive' employment practices had become -at least inside IBM- the only accepted way; a legacy that was taken for granted by both the employees and employers. Of course its contributions to loyalty, morale and other issues were recognized; but those concerns found meaning within a cultural framework in which corporate-based community building was as important as business results.<sup>44</sup>

What is notable is that until the early 1980s the value of employment security was almost never questioned. As I mentioned in the previous section, ever since the Second World War, job (or employment) security had become a part of an institutionalized set of employment practices in the U.S. economy. Even in California (which is often singled out for its mobile job market) companies like Intel, Texas Instruments, AMD and Motorola<sup>45</sup> had employment security policies in place until the early 1980s (Loseby, 1992; Saxenian, 1994). Although companies could not always keep their promises, they saw employment security as the 'progressive' way of organizing the workforce or at least, keeping the unions away.<sup>46</sup> The rise of IBM to the rank of most powerful high-tech company in the U.S. was seen as further manifestation of their effectiveness.<sup>47</sup> Thus, IBM-like employment relationships were considered both as the appropriate and the productive arrangement for organizing human resources.<sup>48</sup>

---

<sup>44</sup> See the incidence reported in Fishman (1981: 89) about the departure of an IBM employee. Compare with one reported in Rogers and Learsen (1984: 150) about layoffs in Intel.

<sup>45</sup> In the aftermath of recession of 1974, Motorola's managers visited HP and IBM to find out about the mechanisms that these companies used to stabilize their employment relationships.

<sup>46</sup> A Fortune magazine survey in the early 1980s found that 1.5% of Fortune 1000 offered employment job security.

<sup>47</sup> Even when Japanese competition began to alert US companies their attention was almost immediately drawn to its similar features like lifetime employment (e.g. Ouchi, 1981).

<sup>48</sup> I do not mean to romanticize IBM's employment relationship. It was at the time criticized for being overly paternalistic. Even though unions were 'unheard of' during much of the history of IBM, dissenting voices among workers existed at least since 1976 (see Early and Wilson, 1986; Alliance@IBM, 2014, cf. Shay, 2012: 218-236). Some white-collar workers also reported that they were 'forced to resign' (see Drandell, 1990). But by all accounts, these were minority voices. <http://www.endicottalliance.org/ibmwuflers/index.htm>

Thus, an ex-post rationalization of IBM's policy can be interpreted as re-appropriation of the function that those legacy practices performed. When IBM was successful they were taken as the key elements in success (evident in the volume of books that were written about it in the 1970s and 80s). When it started to lose, as we see in the next chapter, the same factors were seen as ineffective legacies that only slowed down the business. Indeed, for a brief period in the mid-1980s there was a surge in the volume of books and reports that sought to demonstrate that employment security is beneficial for business (e.g. Mills, 1985; see also Loseby, 1992).

Whatever the intentions and outcomes of IBM's strategy, the decisions that companies made in this context had important implications for the formation of skills in the industry as a whole. A company that chose to uphold its 'traditional' values and maintain its commitments did so by extensive retraining and reshuffling of its workforce. It also had to make sure that the skilled workforce that it trained was productive and contributed to the business. This is the route that IBM chose over the subsequent few years. Others, mostly abandoned any aspiration in this direction and instead adopted new models that were similar to the so-called "Apple's deal".<sup>49</sup> According to Cappelli (1999), these new deals made it clear that employers demanded 'skills' from their employees but they were no longer willing to identify or develop those skills for individuals. This did not necessarily mean that the company would not invest in new skills or will not facilitate learning of skills by individuals. Rather, it means that it can divest the 'skilled' individuals at any point without any further obligation.

Therefore, employees were increasingly encouraged to "direct their attention for career management outside the firm, to the market..." (P.28). In this new formulation, individuals were responsible for their employability that they could secure through 'learning'. Software workers did not feel particularly threatened by this trend, perhaps because the job market had always provided employment opportunities.<sup>50</sup> Thus, while companies almost completely withdrew from providing any training that necessitated employment responsibility and even reduced the training provided to their own employees, software workers failed to form any effective occupational collectivity that protects either their skills or their employment privileges. In the next section, I discuss these issues in the context of labor market of software work.

---

<sup>49</sup> Of course this included many other industries beyond IT.

<sup>50</sup> Burton, K (1985) Beating layoffs a matter of skills, *Computerworld* 8 Jul 1985

Ostendorf, B. (1981) Computer programmers can count on getting jobs, 15 Jun 1981

## 7.5. Analysis of the labor market

### 7.5.1. *Continual problematic of skills*

At the beginning of the 1970s, while the issue of skills was still unresolved, new workers were entering the labor market at an ever-growing pace. As early as 1968, members of ACM Special Interest Group on Computer Personnel Research had warned that there was a growing oversupply of computer workers: “The ranks of the computer world are being swelled by growing hordes of programmers, systems analysts and related personnel. Educational, performance and professional standards are virtually nonexistent and confusion grows rampant in selecting, training, and assigning people to do jobs” (quoted in Ensmenger, 2010a, 18). In 1970, at the dawn of industry’s first recession, an ACM ad-hoc committee on vocational training providers in data processing found that less than 60 percent of ‘EDP school’ graduates could find a job in data processing. While it is not clear to what extent this was due to companies’ ‘no EDP school’ policy, or rise in experience requirements or the looming recession, there was a general feeling that programmers were not adequately skilled.<sup>51</sup>

On the other hand, computer science departments had grown rapidly since the early 1960s and by 1969 about hundred universities in the US provided undergraduate degrees in computer science (see Ceruzzi, 2003: 103; Aspray, 2004). Still, in 1976, a report -sponsored by SHARE and produced by a team from IBM, SDC and a number of user organizations- complained about the status of education in the data processing industry. In their view, there was an overemphasis on “academic respectability” on the part of computer science programs and a clear lack of “engineering-like and management-like disciplines in data processing”. They argued that, due to the lack of this basis, in effect, “programming remains a craft that many people can learn regardless of background, education, experience, etc. but that most will never learn well”. (Doletta, et al. 1976: 24)

The situation was not any better for practitioner-oriented certifications either. Industry’s most well-known certification, DPMA’s Certified Data Processor, had less than 3000 applicants a year. In 1973, perhaps because of a sense of desperation and in the hope that wider support will lead to wider recognition and enrolment, DPMA agreed to transfer the

---

<sup>51</sup> For instance, in the same year, a *Computerworld* survey of job markets in several western cities (including, Los Angeles, San Francisco, Pheonix, etc.) had found that there was almost no job opportunity for “programmers with one year and less experiene” because employers had raised their experience requirements. See Huggins, P (1970) Inexperienced programmers find western job market evaporating, 3 Jun 1970.

responsibility for its exams to a larger body (called Institute for Certification of Computer Professionals, ICCP). This body had been formed by an alliance of no less than eight computing societies, each of which had their own strongly held views.<sup>52</sup>

The *Computerworld* magazine described the process of formation of ICCP as “a scenario which had all the markings of a Grade B political intrigue movie, complete with political struggles, ousters and financial problems”.<sup>53</sup> It is no wonder, therefore, that its establishment did not lead different associations to abandon their rivalries and mutual feelings of distrust. They still failed to agree over what constituted the essential knowledge or qualification in any of the sub-categories of software work (programming, systems analysis, software design, etc.). In 1975, the CDP covered such diverse topics as data processing equipment, computer programming and software, principles of management, quantitative methods and, systems analysis and design. Yet, many found the exam objectionable and unhelpful. The chairman of Association of Computer Programmers and Analysts at the fifth annual conference of the association said that sitting for a CDP exam is “a personal ego trip” for some people. According to *Computerworld* reporter, the attendants of the conference agree that some sort of knowledge test is necessary, “but [the] agreement ends there”.<sup>54</sup> Similarly, ICCP’s new certificate, called the Certificate in Computer Programming (CCP), failed to make an impact while the number of people taking CDP fell even more. Some, including members of the ICCP’s constituting body, accused ICCP of creating the new certificate with the purpose of undermining CDP. ICCP’s defence was that CCP is “targeted at upper skill levels of those practicing business, scientific or systems programming”<sup>55</sup>. But when there was little agreement about the basic qualifications, there could hardly be any agreement about the higher levels.

### ***7.5.2. Status of programmers according to the courts and government***

In November 1970, the US Department of Labor initiated hearings into the status of computer-related occupations and whether or not they should be exempt from overtime provisions of the Fair Labor Standards Act of 1938. The main industry group present at the hearings was the Association of Data Processing Service Organizations (ADAPSO), which

---

<sup>52</sup> Beyond the DPMA and ACM, they included Association of Computer Programmers and Analysts, the Association for Educational Data Systems, the Automation One Association, the Society of Certified Data Processors (SCDP) and the Society of Professional Data Processors. The IEEE (Institute of Electrical and Electronics Engineers) Computer Society and the Canadian Information Processing Society joined later.

<sup>53</sup> *Computerworld* (1974) ‘Saga of ICCP has happy end’ 2 Jan 1974.

<sup>54</sup> Bride, E.J. (1975) Test of DP knowledge seen necessary, 17 Oct 1975.

<sup>55</sup> Brown, J. (1975) ‘CCP won’t replace CDP: Articles on certification rife with paranoia’, *Computerworld*, 1 Aug 1977.

could save huge amounts from not having to pay overtime wages<sup>56</sup>. One central subject of the hearings was the status of programmers: if they were considered 'professional' employees they would be exempt from overtime pay. In order to be considered a professional job, a job in any occupation had to 1) require at least a college degree as its entry requirement, and 2) exercise discretion and independent judgment.

ADAPSO representatives argued that the first requirement could be substituted by post-secondary technical course, on-the-job training and work experience. Employee representatives, on the other hand, refused to be recognized as a profession without having satisfied the credentials required. About the second requirement, the spokesman argued that programmer is "the captain of a ship... required to make instant decisions."<sup>57</sup> For him, "the crux of the problem" was "definitions". Ultimately, the hearing committee was not satisfied. Its decision regarding whether programmers and systems analysts could be included in "the learned professions" was that:

"At the present time, there is too great a variation in standards and academic requirements to conclude that employees employed in such occupations are a part of a true profession recognized as such by the academic community with universally accepted standards for employment in the field. Some computer programmers and systems analysts may have managerial and administrative duties which may qualify them for exemption." (Quoted in Willoughby, 1975: 10)

Nevertheless, it concluded that if jobs of specific "data processing" personnel involved other activities (like managerial tasks) or other forms of "discretion and independent judgement", they could be considered professionals (and exempt). The decision also set out the criteria that should be used to classify programmers and systems analysts, but still left too much room for interpretation.<sup>58</sup> The result was that over the subsequent years, several employees, who insisted that their jobs did not involve "exercising independent judgment", took their employers to the court.<sup>59</sup> Judgments varied from case to case, and the general issue remained unresolved and confusing throughout the period.

---

<sup>56</sup> Another aspect of Fair Labor Standards Act concerned minimum wage which was not generally an issue.

<sup>57</sup> Quoted in *Computerworld* (1976) Editorial: Time to face facts, 16 Feb 1976.

<sup>58</sup> "Examples of work *not* requiring the level of discretion and judgment contemplated by the regulations are highly technical and mechanical operations such as the preparation of a flow chart or diagram showing the order in which the computer must perform each operation, the preparation of instructions to the console operator who runs the computer or the actual running of the computer by the programmer, and the debugging of a program." (Quoted in Willoughby, 1975: 11)

<sup>59</sup> Arnst, C (1976) 'Called nonprofessionals: programmers win overtime', 9 Feb 1976.

On a separate but related issue, in March 1971, in a widely publicized case between the National Labor Relations Board and Westinghouse Electric Company, the U.S. Court of Appeal ruled that programmers must be categorized as ‘technical’ rather than ‘professional’ employees. The main issue at hand in that case was whether programmers and systems analysts could be union members and the company was arguing that they were professionals who could not be union members. The judge ruled that the jobs in question did not require “knowledge of an advanced type” and therefore, could not be regarded as professional. In his ruling, he also pointed out that only two of the six programmers in that particular case had a college degree and, furthermore, four of the six were previously members of the unionized sections of the company.

It seems that despite these measures, programmers and other computer workers avoided unionization. The only major early attempt to create an industry-wide union, which was initiated in December 1970, never took off. The Committee to Plan a Computer Union was formed by 75 programmers and keypunch and computer operators who attended the first meeting in New York.<sup>60</sup> Their objective was to bridge the gap between “professionals (programmers/analysts/operators) and non-professionals (keypunch-operators/ tape-handlers)” and to promote “job security, job mobility and democracy in the workplace”. But by November, 1971, they had realized that there was no real prospect of mobilizing computer workers (Garner, 1974). Moreover programmers actively tried to dissociate themselves from the unions, making the headlines several times during the 1970s.

### ***7.5.3. SCDP and the licensing of Data Professionals***

Disappointed by the limped movement of ICCP, a grassroots organization of CDP holders, called the Society of Certified Data Professionals (SCDP), decided to take the issue to a different level by bringing in state licensure. For years, CDP holders had argued on the pages of *Computerworld* that a ‘CPA-like’ certificate was necessary for the field of computing. But dissident voices had always responded that the purpose and methods of accounting are different from data processing and therefore, the comparison was misplaced. Eventually, SCDP members gave up on trying to create a consensus and decided to take matters into their own hands. In December 1974 they submitted a draft to state legislatures across the country that, if passed, would make practicing in the computer labor market subject to being licensed by the state.

---

<sup>60</sup> *Computerworld* (1970) ‘Computer union group takes action’, 23 Dec. 1970.



The proposed “Draft of Legislation to License Data Processing Professionals” was a concrete attempt at closing the gates of computer labor market. According to the proposed bill, only persons who had either 1) a four-year degree in data processing and three years of related experience, or 2) five years of experience and passed a certification exam could “practice, continue to practice, offer or attempt to practice data processing or any branch thereof”. A ‘blanket approach’ was taken in specifying the job titles that were to be covered by the bill including those that contained terms such as “data processing” and “computer professional” and any of the like. Boards of registration were to be set up to evaluate applicants. Those who had at least twelve years of experience could be exempt from these requirements, contingent upon registering with the state within 5 years. Others were given two years time to acquire the necessary qualifications.

The board was also authorized to revoke the license of any practitioner if he or she was proved guilty of acting in negligence, violating the professional code of ethics or committing fraud. For Kensington Lord, president of SCDP, this was the most important aspect of the bill:

“One does not truly have a profession until one has the ability, legally, to challenge a practitioner and when proven guilty, to see that he is separated from the practice. ... This is one problem that the SCDP bill will solve.” (Quoted in Ensmenger, 2010a: 182)

Unsurprisingly, bitter controversy ensued the news that SCDP had submitted the proposed bill. The Association of Computer Programmers and Analysts (ACPA) released a statement, announcing that while they continue to support “comprehensive, voluntary certification programs”, they “actively oppose any proposal supporting mandatory licensing which would limit the individual practitioner’s freedom to practice” his profession. Interestingly, they also warned that success of the bill “could well evolve into complete national unionization of various levels of DP practitioners” something that posed “a real threat to areas of this country”.<sup>61</sup> In a survey of Computerworld readers, 72% of respondents voted against licensing.<sup>62</sup> An opinion piece appearing in Computerworld called directly on data processors to oppose the licensing scheme by writing to their legislators: “for your own protection of your rights of freedom, to gain your livelihood and your ambition without interference and harrassment from unqualified government agencies”. He warned that legislation would restrict job choice for individuals by making “acceptability as a DPer” determined “by law (not ability)”. This would create a sub-class of non-qualified employees

---

<sup>61</sup> *Computerworld* (1975) Stronger stand urged to defeat SCDP bill, 21 May 1975.

<sup>62</sup> Arnst, C. (1975) ‘SDCP licensing proposal splits community in 1975’, 31 Dec 1975

who would form a union to gain recognition and therefore politicize the working environment.<sup>63</sup>

These discussions also brought to the surface the fears and reservations that some programmers had about unionization. Both the ACPA statement and the letter published in *ComputerWorld* warned against the implications of licensing for formation of unions among programmers. According to ACPA, mandatory licensing “could very well evolve into complete national unionization of various levels of DP practitioners”, which would “become a real threat to this country”.<sup>64</sup> The industry -which had largely remained silent on the discussions of professionalism- voiced its concern. Oliver Smoot, vice president of the Computer and Business Equipment Manufacturers Association (later Information Technology Industry Council), told the attendants of DPMA conference that he opposed the idea of licensing. While recognized the increasing ‘public pressure’ for computer workers to become a profession, he argued that licensing is not appropriate because “data processing is not a profession in the same sense as law or medicine”. Instead, he defined professionalism “as exposing yourself to personal responsibility”, especially against clients. He also emphasized that professionals should have “almost complete control” over the work they do and suggested that practitioners push for “better educational opportunities and enforcement of ethical standards”.<sup>65</sup>

I have not found any other evidence of intervention or lobbying by the industry, but it is almost clear that there was no need for serious lobbying. Only three states took any action regarding the proposed bill: one (Massachusetts) rejected it and two other (Florida and New Hampshire) never enforced it. Even those who were sympathetic to the cause felt that the bill had appeared “too soon”. But the passage of time only led to the bill dying a quiet death. During the rest of the 1970s, discussions about licensure died down. ICCP remained divided and IT labor market continued to expand unchecked.

## **7.6. Implications for software work**

In this chapter, I have shown that competition was transformed by the arrival of PC. It is often pointed out that the success of personal computers was largely due to IBM’s open approach in its design and development of IBM PC (e.g. Langlois, 1992). I have tried to illustrate that almost all of the earliest entrepreneurs in the new PC software market had built on their experience in the mainframe era as well as the skills that had become widely

---

<sup>63</sup> Hamilton R. (1976) ‘Licensing proponents ‘cashing in on calamity’, *ComputerWorld*, 16 Feb, 1986.

<sup>64</sup> *Computerworld* (1975) ‘Stronger stand urged to defeat SCDP bills’, 21 May 1975.

<sup>65</sup> Arnst C. (1975) ‘CBEMA official sees responsibility as key to professionalism, 16 Jul 1975.

available in that period. In doing so they also created a new industry that was separate from the mainstream computer industry not only in its business models but also in their employment relationships. This created a variety of conditions for software work that ranged from job hopping between various software companies to long-term employment at one company. While this variety could be probably found in the mainframe era as well (especially among employees of corporate software products), divisions in the structure of industry as well as rapid growth of new PC software companies made the newer forms of software work more visible. Still the new employment relationship had not been institutionalized and many employers were trying to keep their skilled software workers. On the other hand the labor market continued to expand without any governmental or occupational control. Professional associations were divided and fought each other, weakening the position of all of them. As I explore in the next section, when the new employment relationship became widespread and institutionalized, there was almost no occupational collectivity that could negotiate the new deal or create barriers for entry into the labor market.

## **8 New competition, new workplace: 1985-2001**

### **8.1. The rise of Microsoft**

The rise of personal computers as a mass market for the computer industry transformed the shape of competition during the late 1980s and early 1990s. The first signs of a changing market arrived in 1986, when it became apparent that IBM's earnings for 1985 had fallen by 10%. In the same year, Burroughs acquired Sperry and was renamed Unisys. CDC's revenues more than halved between 1988 and 1990 and the company ceased to exist in 1992, when it was split into two companies. Honeywell, which had entered joint ventures with a number of non-American companies during the decade, finally sold its computer division to the French company Bull. NCR was taken over and became a subsidiary of AT&T in 1991 and was later renamed as AT&T Global Information Systems. Thus by early 1990s, all of IBM's original competitors (the BUNCH) had either disappeared or were struggling for survival. IBM alone had nearly 90 percent of the mainframe market.

But IBM was also struggling in the second half of 1980s. Its mainframe business had stalled and was facing fierce competition from Japanese companies (notably, Hitachi). It had also lost much of its share of the PC market, which was the fastest growing segment of the industry. At the same time, no company in the PC industry had a dominant share of the market. As early as 1985, Compaq, AT&T, HP (from inside the U.S.) and Fujitsu, Olivetti, NEC, Panasonic, and Toshiba (from outside) had been manufacturing PCs, while others (like Dell) were using innovative business models to assemble and deliver customized PCs. IBM tried to regain its position in the PC market by separating its way from the rest of industry. As part of its effort to differentiate itself, IBM developed a new product line called Personal Systems (or PS/2) and, in collaboration with Microsoft, a new operating system (called OS/2). When the new line of products was introduced in 1987, it failed to impress the market and resulted in IBM looking out of touch with the demands and direction of the market. In 1988, IBM's personal computer division lost more than one billion dollar. By 1990, its share of the PC market had been reduced to 14% (Evans, et al, 2006: 92).

Another development with roots in the first half of 1980s that damaged IBM's position was the advance of workstations. These desktop computers were more powerful than PCs but cheaper than minicomputers and could be used up with each other. Although primarily directed towards scientific and technical applications, workstations' success came at the expense of mainframes because they inhibited the growth of markets for mainframes. Two of the foremost vendors of workstations were Apollo Computer (later acquired by HP) and

Sun, both of which used the freely-available UNIX operating system. DEC, on the other hand, developed workstations that used its proprietary operating system VMS and could be networked with its successful line of minicomputers. This strategy was so successful, it made DEC the second most successful computer vendor in the industry. DEC also expanded its software activities and became a major provider of programming services by developing custom-made applications for its customers. But ultimately it was Unix-based systems that won the competition and pushed DEC aside.<sup>1</sup>

At the same time, since a variety of computers (ranging from mainframes to PCs and minicomputers to workstations) could be found in any corporation, IT companies began to develop networking technologies that enabled different computers to communicate with each other. While some vendors had begun such efforts as early as 1960s and most of the advances in these technologies were made in 1970s, it was in the 1980s that these technologies found a growing market. For example, most of the workstations used Ethernet technology, which was developed in Xerox-PARC in mid 1970s. Ethernet was later used to connect PC computers as well. Gradually, certain companies (such as 3Com and Cisco) concentrated on manufacturing networking devices, forming a separate sector in the structure of industry.

As a result of these developments, the structure of computer industry significantly changed. In the new landscape, no single corporation had absolute dominance in terms of market share. Instead, its influence was based on the extent to which it could control one or another layer of technology in common use. Thus, as IBM's power was fading, Microsoft was emerging as the new powerful company because it had the most 'installed base' in the PC market. By all accounts, about 90% of the new PCs had MS-DOS as their installed operating system.<sup>2</sup> Although initially Microsoft's direct income from licensing of MS-DOS was small, by 1986 MS-DOS was generating almost half of Microsoft's revenue (Campbell-Kelly, 2003: 242). Microsoft also expanded its support for BASIC and C programming languages and started working on Windows operating system that ultimately replaced DOS as *de facto* operating system of the PC market (see below). By 1995, Microsoft had 50 percent of the PC software industry but the overall structure of industry had remained the same: Microsoft and a dozen other companies accounted for more than three quarter of the market. Still, in

---

<sup>1</sup> The success of DEC was short-lived. The company started to make losses in 1992. After several rounds of layoff and restructuring, the company was sold to Compaq in 1998. By that time, its number of employees had reached 53,000 (from 126,000 in 1989).

<sup>2</sup> Campbell-Kelly (2003: 241) names more than twenty operating systems that were competing with MS-DOS in 1985. Microsoft itself was initially focused on developing a UNIX-based operating system called XENIX but later decided to abandon it.

terms of the overall software industry (PC and corporate sectors) Microsoft's share was less than 10% (Campbell-Kelly, 2003: 26).

Novell was another company that used a strategy similar to Intel and Microsoft; that is, reaching a wide installation-base while enabling more applications to be run. By developing a network software that laid upon MS-DOS, Novell became the most prominent provider of networking software in the PC industry. By 1995, it had become PC software industry's number two in terms of revenue (Campbell-Kelly, 2003: 235). Borland, similarly, was successful because its programming language (Pascal) was used to develop many other applications. Thus, the second half of 1980s witnessed the rise of successful PC software companies that their products were not intended for end-users but rather served as enabling layer for further applications. In comparison, none of the top corporate software companies were developing operating systems or programming languages. They were either providing a range of software products (like Computer Associates) or a highly successful database management systems (e.g. Oracle, Sybase). There were providers of programming aids ('application generators' and 'program generators') but most of them only enjoyed a modest success. Therefore, there was no similar layered structure in the corporate software industry. The only software that was the basis of development of further applications in the corporate software market was IBM's CICS (see Campbell-Kelly, 2003: 149-152).

Meanwhile, after a false start with Windows, in 1988 Microsoft released Windows 2 which sold two million copies in two years (Campbell-Kelly, et al., 2013). Microsoft's Windows 3.0, which was released in 1990, was even a bigger success, selling ten million copies in two years and replacing MS-DOS as the industry standard (OS/2, in comparison, sold only an average of 100,000 copies each year, Levis, 2009). Moreover, Microsoft turned its installed base to an income source by developing a range of office productivity applications. Microsoft's chief competitors (Lotus and WordPerfect) had invested in developing an OS/2 based software and therefore, could not compete with Microsoft on its Windows platform. Soon, Microsoft's Excel and Word became the most widely used office applications. Wintel (a combination of Intel processors and Windows operating system) replaced "IBM PC-compatible" as the defining feature of the personal computer market. Moreover, by enhancing its network capabilities (especially in Windows 1995 and NT), Microsoft was able to capture a large proportion of Novell's market share.

Taking drastic measures to rescue itself, in 1986 IBM began an extensive restructuring program which was not completed until 1994. During this period it reduced its workforce by more than 45%. IBM's annual revenues dropped year by year between 1991 and 1993,

reaching a record loss of \$8 billion in the first half of 1993. After recovery, IBM refocused its market strategy on software and services (for example by acquiring Lotus, the PC software company in 1995). Moreover, IBM embraced the open architecture on its various systems by using industry standards that would make them compatible with various hardware and software technologies. In 1991, it formed an alliance with Microsoft and Intel's main competitors, Apple<sup>3</sup> and Motorola, to undermine the influence of Wintel standard in the industry. The alliance, however, failed to make any impact on the market and its only outcome was a series of microprocessors that IBM and Apple used separately on some of their products.

Other mainframe and minicomputer vendors implemented similar restructuring plans. Unisys, which in 1986 was the second largest seller of software (after IBM), moved into workstation and PC markets in the late 1980s. Its survival was, however, still at risk. Between 1989 and 1992, Unisys underwent a major restructuring effort during which more than 50,000 employees were laid off (making the size of company less than half of what it was at the time of merger).<sup>4</sup> It also established a new services division to provide consultancy and system design and integration services. By 1995, it had more than 7,000 employees providing such services.<sup>5</sup> But after a short period of recovery, Unisys still faced difficulties and undertook several more rounds of restructuring in the late 1990s. Similarly, NCR was making losses throughout the most of 1990s and was spun off by AT&T in 1997. After becoming independent, NCR too restructured its business processes and cut staff. The renewed company concentrated on transaction processing systems (e.g. ATMs) and related services (such as data warehousing).<sup>6</sup>

While these companies were struggling to reposition themselves in the industry, the surge in using PCs at home during the mid 1980s and early 1990s also created a vast consumer base for various forms of information services. The pioneering firm in this sector was CompuServe, which operated as a mediator between computer users as well as content providers (like newspapers and magazines) and consumers. Soon other companies like America Online (AOL) began to create their own proprietary information services. In these networks, consumers connected to the servers of these companies to access the information provided by content-providers or communicate with other subscribers. The business model

---

<sup>3</sup> After overcoming the difficult years of mid 1980s, Apple had a comeback with Macintosh computers.

<sup>4</sup> In 1986, the combined headcount of Burroughs and Sperry was around 120,000. 10,000 workers were laid off shortly after the merger.

<sup>5</sup> *InfoWorld* (1995) 'Consulting companies offer different strengths' *InfoWorld*, 7 Aug 1995.

<sup>6</sup> While still a division of AT&T, NCR had left the PC market in 1994.

of access-providers was based on subscription fees from consumers and server charges to content providers, but their networks were closed to anyone who was not a paying member.

At the same time, expansion of the Internet had made it possible for computers to communicate without any limit as long as they could join one of the networks that was connected to the Internet. Internet Service Providers (ISPs), provided access but users were responsible for finding relevant content and/or communication tools.<sup>7</sup> By mid 1990s, the Internet had become an attractive application domain. Especially, popularity of the World-Wide Web (or the web, for short) as a means of providing content attracted several companies (like Netscape, but also IBM and later Microsoft) to develop browsing applications for customers. These companies operated on a different business model: ISPs were only access providers and browsers were applications that enabled users to surf the web. Servers, on the other hand, were independent and could be set up by anyone who had the server software installed.

In the early 1990s, Microsoft decided to set up its own proprietary information system and bundled it with Windows 1995. But by then Netscape had grown to become the dominant player in the browser market, having more than 70 percent of the market. In a tactical move, Microsoft tried to push out Netscape by providing access option to AOL services through its Internet Explorer application on its Win32 platform, even though it meant sacrificing Microsoft's own network (MSN). Microsoft's Internet Explorer, which was included for free on Windows operating system, provided access to both AOL's services as well as the web. This resulted in the famous 'browser wars' between Microsoft and Netscape which provoked the US Department of Justice to file an antitrust suit against Microsoft's bundling practices in 1998. The verdict, handed down in 2000, was that Microsoft was guilty and should be broken up. But following Microsoft's appeal, the original judgment was set aside and replaced by less drastic measures.

In the mean time, Microsoft had grown enormously earning \$23 billion in 2000. AOL, which was endorsed by Microsoft, also gained a seemingly unstoppable momentum, acquiring CompuServe in 1998 and Netscape in 1999. By 2000 it had become the *de facto* gateway to the Internet, surpassing all other ISPs. The increasing interest in the Internet that developed from the mid 1990s onwards was to a large extent the result of gradual withdrawal of government support for its backbone network (a phenomenon known as the

---

<sup>7</sup> Internet is often seen as an outgrowth of the ARPANET, but data processing organizations played an equally important role in development of the concept of networking and its technologies (see Abbate, 2000; Campbell-kelly and Garcia-Swartz, 2013). Development of data communication lines can be traced as far back as the SAGE project.



privatization of the Internet). But arguably, privatization did not have as much impact as the commercialization that followed. While the Internet was originally developed as a communication and collaboration tool between scientists in universities and research centers, in the mid-1990s it had become an open informal network that anyone was free to join and was mainly used for non-commercial purposes. By the late 1990s this had changed and a frenzy of online activity was taking place inside American corporations.

While initially most of these development efforts were simply aimed at 'web presence', later more sophisticated efforts were directed at developing e-commerce systems both among companies and between companies and users. One of the tools that facilitated this rapid development of web-based applications was Sun's Java programming language, which was made available in 1995. Java was highly suited to web applications because it enabled programmers to write code once and run it anywhere (that is, almost on any computer). Soon the web was the new domain of both business and application development for companies. Some like Yahoo (founded in 1994) were oriented towards facilitating access to the resources available on the web (by indexing in the case of Yahoo and crawler-based search in the case of Google). These companies' main source of revenue was advertisement. Other early successful ventures (e-businesses) were Amazon and eBay. The success of these businesses fed into the frenetic development of e-commerce applications and e-business tools with huge expectations. When many of those business models failed to succeed and the euphoria of Internet-based glories passed, the stock value of most prominent e-businesses crashed. The dotcom bubble had burst.

## **8.2 Analysis of the competition**

In discussing the causes of IBM's fall from its dominant position, often too much credit is given to the rise of PC and Microsoft's clever strategy (especially its pricing of MS-DOS and the fact that it maintained the rights to it rather than just selling it to IBM). It must be noted, however, that IBM's revenue from its PC division compared to its total revenue was relatively small. The drop in IBM's share of the PC market could hardly cause a serious financial problem. What seems to be the main cause of IBM's decline is the implications of growth in IBM PC and its clones for IBM's mainframes and related products. Therefore, in this section, I briefly consider the internal causes of IBM's downfall before moving to analyzing the effects of competition.

While IBM generally underestimated the popularity of PCs and the extent to which it will affect the industry, it seems to have also overlooked the immediate impact of PCs on its mainframe business. This overlook occurred, at least in part, because in 1980 IBM changed

its mainframe sales strategy, advising its customers to buy rather than lease computers. This created a period of income growth that deluded IBM managers into believing that the mainframe market will be continuing to grow. Subsequently they made two mistakes: first, they put more people in manufacturing mainframe systems just at the time that the PC market was on the rise. Second, by switching customers to purchasing instead of leasing, they disrupted the historically formed relationship between IBM sales personnel and its customers. This weakened the importance of sale and support personnel even more (it was already undermined by distribution of PC through alternative channels). Changing the relationship between the sales personnel and customers further not only took away one of IBM's greatest sources of advantage, it disconnected IBM from its customers. When the additional income from conversion to ownership slowed, IBM's income revenue dropped sharply because it had much fewer computers on lease.

On the other hand, the open architecture of PC not only made competition possible but also gave rise to a new form of competition. The vertical disintegration that began in the early years of 1980s took shape in the second half of 1980s and early 1990s. In the new vertically disintegrated industry, the distinction is no longer simply between hardware and software but between various layers of software and hardware that work together. Therefore, rather than competing for delivering total systems, in the so-called platform competition (discussed in chapter 3), different companies try to use their position as provider of one or another layer of technology to influence others. Microsoft did so by having its office productivity applications ready for the new Windows platform, and then bundling Internet Explorer with it (when web-surfing became an important domain). Moreover, Microsoft tried to expand its dominance in the market by not only encouraging other companies and developers to develop products for its operating systems, but also supporting them (by providing them with Application-Program Interface training and tools like 'developers kit').

Although no other company was as successful as Microsoft, some others did try to achieve a similar form of dominance. For instance, at the networking layer, Novell used its software to dominate intra-organizational networks. Later Cisco used its networking technologies initially to provide solutions for intra-organizational networks but later became an important provider of components used in the Internet infrastructure by ISPs and others. AOL, similarly, first dominated the market for information services but later became a dominant ISP. On the other hand, some companies like IBM and Sun were involved at several layers of technology at the same time. Thus Sun remained a major provider of workstations at the same time that its Java was dominating the market for programming languages for web-applications. It is notable, in this context that Microsoft tried unsuccessfully to take control

of Java programming language or at least undermine Sun's control over it. Sun took Microsoft to the court for its 'incomplete implementation' of Java and, at the same time, formed a coalition with Novell, IBM and Oracle to prevent Microsoft from overtaking Java. Ultimately, in order to reach a settlement out of court, Microsoft agreed to use Java as specified by Sun in 2001.

Hence, while competition over programming languages had a long history in the computer industry, the main difference between early stages of competition and the period after mid 1980s was the layered structure of industry. Companies had to decide which layer (or layers) to compete in and how to draw on external resources (including other technology providers) to ensure the success of their business models. Partnerships and product/brand licenses became an important feature of the dynamics of industry. However, the new forms of competition required both making decisions about product architecture and relationships with other vendors (as Gawer and Cusumano, 2002 and Evans et al., 2006 have shown) and ensuring that competent human resources existed to enable those technologies. Therefore, IT companies had an interest in propagating the ideas and visions behind their technologies and disseminating the skills required for their further development and application through training. At the same time, employment relationships in the industry were undergoing a sea change that had similarly significant implications for how training was provided. Therefore before considering the training policies of computer companies and the new institutional form that it took, I will discuss changes in the employment relationships in the industry.

### **8.3 Employment Policies**

In the previous chapter, I described the varieties of software work that existed in the industry. I showed that for a period of time, two conflicting norms of employment were operational in the industry: one that was committed to employment security and one that encouraged employees to take responsibility for their employability. As the old model of dominance was being replaced with new one -and IBM with Microsoft- the norms of employment were also being transformed. In this section I discuss how the old norms of employment disappeared and new norms came to dominate employment relationships in the industry. I will start by discussing the transformation of employment relationships in IBM, which is perhaps the best illustration of differences between the old and the new norms in practice. Then I will discuss Microsoft's employment relationships as an example of the new norms.

### ***8.3.1 Employment policies and decisions in IBM***

In the second half of 1980s, when layoffs were becoming increasingly the norm, IBM tried to avoid layoffs<sup>8</sup>. Between 1985 and 1986, it spent \$550 million to retrain its employees across the organization. By 1986, when it had completed its retraining and relocating efforts, it had reassigned 21,500 of its 273,000 U.S. employees into new positions, training 10,000 of them for new jobs. Among them, there were 7,600 employees in marketing and other overhead assignments who had been reassigned as sales representatives or system (or field) engineers. Another 9,600 employees were moved from manufacturing and product development to other sections of the company: 3,200 became sales and marketing support personnel and customer engineers and about 2100 became programmers. Nearly 7,000 employees were physically moved around the country. It also provided a voluntary retirement program through which 13,000 employees chose to leave.

These drastic measures were taken to preserve the competitiveness of company while preserving employment security of employees. Therefore, many of the workers who were not moved continued to work in recommissioned plants that had changed their operations. In one case, in 1986, a manufacturing plant in Boulder (CO) was recommissioned as a programming, distribution and service facility, while its production line was moved to plants in Lexington and Charlotte. Within 8 months, the plant site was redesigned and converted to an office that would be capable of accommodating mainframes (to provide ventilation and cooling). Of the 2,700 workers that were affected by these changes, IBM physically moved more than 600 to locations near the new production lines. 900 remained in their jobs or were assigned to jobs that were similar to their previous jobs. The rest (about 1,200 workers) were retrained to enter entirely new jobs, which were roughly divided into equal number of blue-collar and white-collar jobs. While some blue-collar workers became system engineers (for customer computers) after 3 months of training, a few hundred workers were retrained (in a 15-week program) and employed as programmers. These became part of a new programming and information system support group which was involved in federal government projects.

Moreover, during the same period, a full programming unit (consisting of about 200 programmers) was transferred from Westlake (CA) to Boulder. The Westlake facility was established in 1968 as part of IBM's Federal Service Division to provide services to the nearby manned space laboratory. But just after IBM had hired several hundred programmers for the project, the project was cancelled. Rather than laying off new employees, IBM found

---

<sup>8</sup> This section about IBM's retraining programs draws mainly on (Mills, 1988) and Loseby (1992).

ways to employ them in other projects and nearby locations. Over time, Westlake programming unit had become an established center for providing programming services but because of the ebbs and flows of projects it was always susceptible to fluctuation in its personnel. (In 1972, for instance, 300 employees left when projects at hand were completed but no new project was in sight.) They were also few growth opportunities within the unit because it was disconnected from other projects and activities that took place in Federal Service Division and were mostly concentrated in Washington D.C. area.

Relocating to Boulder affected 455 employees but less than half (222) were physically moved to Boulder. 97 employees chose to be relocated in the LA area or transferred elsewhere. The manager in charge of Westlake unit convinced about 100 of those who did not want to move to Boulder to take temporary assignments there. This was necessary because the ongoing work in Westlake had to be transferred to Boulder and not only unfinished tasks had to be completed, but also new programmers had to be trained and new work units had to be set up. Nevertheless, during this process of relocation, 82 employees used the early retirement opportunity to leave the company while 54 others resigned.<sup>9</sup> In July 1987, the total number of employees in the newly set up Federal Service Division programming unit in Boulder (including relocated Westlake personnel and reassigned Boulder personnel) was near 600. According to the existing reports, this relocation effort did not create any significant delay or major difficulty in the projects.

In another case, a manufacturing plant in Greencastle (IN) -built in 1953- was recommissioned in 1975-6 as a distribution center. But in the mid 1980 it had lost its position in the distribution channel and there was no longer any economic justification for its operation. Almost 1000 employees were affected with an average of 22 years of employment at IBM, more than half becoming eligible for retirement in 1986. The plant was closed in March 1987. Those who retired or chose to leave received assistance for finding new jobs, \$5000 as retraining assistance, and two-years extra payment for increased commuting costs. All remaining employees (including those who did not wish to retire) were offered jobs in nearby locations and moving assistance from IBM. Many production workers took their retraining assistance packages to become programmers.

Most of the people were, naturally, excited about (and grateful for) the efforts that the company was putting in retraining them:

---

<sup>9</sup> Carroll (1993: 160) maintains that the voluntary retirement packages were not available to those in programming because they were highly in demand. But the Westlake case suggests that this is not true or at least such a rule was not applied everywhere. In any case, programmers could always forego the retirement benefits and just resign.

“My shift to programming has been a benefit to me and to the company. I looked to see how it would benefit me first. I decided that if the company is willing to educate me, what do I have to lose?...I went through two classes with twenty-five other people. We got to be like a close-knit group. We cared about each other-like a family.”

Nevertheless, redeployment of people created some awkward situations. One in 8 managers had moved to new fields in which they had no experience. A manager of the manufacturing plant who had been reassigned to a programming group reportedly introduced himself saying: “I’m your new manager. Would you please tell me what you do so that I can be of some assistance to you”. In his account of the fall of IBM, Paul Carroll (1993: 102) reports that IBM used programming “as a dumping ground for people in IBM plants whose jobs had been eliminated”. According to him, Microsoft programmers participating in IBM-Microsoft joint projects in the mid 1980s were “shocked” to see retrained programmers assigned by IBM.

But IBM was proud of its continual commitment to employment security. As indicated above, when a plant or office was closed, employees were offered employment options in other locations. Through its early retirement and voluntary layoff options, IBM had managed to reduce the total (worldwide) number of its employees from 405,500 (its peak in 1985) to less than 374,000 in 1990, without ‘firing’ anybody. In an IBM management report in September 1987, IBM’s vice president of Personnel, Walton Burdick was still able to write:

“While other companies responded to industry slowdowns with layoffs- and so were able to improve short-term financial results quickly- IBM continued its no-layoff practice. I think we[IBM]’ll reap long-term benefits through our most important asset- people.”

Towards the end of the 1980s, one of IBM’s advertising campaigns was formed around boasting its no layoff policy:

“Over the years we’ve retrained thousands of employees by offering them the opportunity to learn new skills....Retraining has shown us how a company and its employees can change together in a changing world. Above all it has shown us that jobs may come and go. But people shouldn’t.”<sup>10</sup>

Of course, when options seemed irrelevant to employees’ current positions or required moving to a faraway location, some employees felt that they were being effectively pushed out. More importantly, it seems that in the first waves of voluntary layoffs, IBM lost many

---

<sup>10</sup> This excerpt is from New Yorker, 27 May 1985, p. 65.

Jack Welsh, the CEO of GE who is often recognized as one of the leading figures in changing the employment relationships in the U.S. referred to this campaign in his autobiography *Jack: Straight from the guts* as a contrast between IBM and his policies at GE.

of its best employees. According to Carroll (1993: 160-61), of the first ten thousand to leave, eight thousand were rated “one”, which constituted more than one third of the whole number of people so-rated in IBM’s U.S. workforce.

By the start of 1990s, IBM’s problems had not been resolved and increasing numbers of employees were actively encouraged to leave. But the real change in IBM’s employment policy occurred in October 1992, when IBM changed its ‘full employment’ policy. While insisting that employment security continued to be a key ‘objective’ for IBM, the new policy made managers of business units responsible for ensuring that no layoffs were required (otherwise, they could lose their jobs). Nevertheless, one implication of the new policy was that involuntary layoffs could now be considered an option. In March 1993, involuntary layoffs began with terminating 2,600 employees. Within a decade from 1986, IBM’s global employment reached 219,800 while the number of its U.S. employees was effectively halved (from 237,000 to 125,000, see table 8-1). With the arrival of Lou Gerstner as CEO in April 1993, employment relationships at IBM were fast tracked into the new order. Gerstner was hired from outside IBM (an unprecedented event in IBM’s history) and saw no value in trying to avoid layoffs. Whereas most of the previous layoffs were occurring in small numbers, more than 35,000 employees were laid off within a year of Gerstner’s arrival.

Year	1986	1989	1996	1997	1998	1999	2000	2001
U.S. Employees	237	223	125	136	147	150	153	152

Table 8-1: Number of IBM U.S. employees (in thousands). Accurate figures for 1990 until 1995 were not available.

Part of the change in IBM’s employment structure was due to its strategy of moving out of hardware and into software and services.<sup>11</sup> As I mentioned in the previous chapter, software activities in IBM were scattered in different departments and locations. For instance, according to Carroll (1993, 188), about 1,700 programmers who worked on OS/2-related projects were distributed between four sites on two continents. Another project (OfficeVision) involved 1,500 programmers distributed across nine sites. By early 1990s, IBM had 30 software laboratories around the country (p.226). Under Gerstner, software laboratories were consolidated into 8 and became part of a unified software division. Since there was no longer any commitment to costly retraining programs or retaining long-time IBMers, IBM managers were free to replace their existing employees with new hires from

<sup>11</sup> Share of revenues from software and services rose under Gerstner from 43% in 1992 to 58% in 2001.

outside. Therefore, as early as 1994, IBM began hiring at the same time that it was still laying off its employees. Between 1995 and 1997, IBM hired and trained 5,000 new software sale specialists (the number grew to 10,000 by the end of decade, Gerstner, 2003: 140-41). At least some of those workers were former IBM employees who had been laid off recently.<sup>12</sup> As early as 1995, one in five of laid off IBM employees were being re-hired as ‘consultants’, IBM’s term for temporary workers (Cappelli, 1999: 74). It was a remarkable sign of change at IBM that when IBM acquired Lotus in 1995, one of the terms of merger (imposed by Lotus) was that no Lotus employee was to be laid off as a result of merger.<sup>13</sup>

What is interesting in this regard is how Gerstner justified and legitimized his radical changes (by IBM standards) for the senior management. By his own account, in 1994, in a meeting with IBM’s top managers he showed them photos of CEOs of their competitors (e.g. Microsoft’s Gates, Sun’s McNealy, Oracle’s Ellison) and told them:

“One hundred and twenty-five thousand IBMers are gone. They lost their jobs. Who did it to them? Was it an act of God? These guys came in and beat us. They took [our] market share away and caused this pain in this company. It wasn’t caused by anybody but people plotting very carefully to rip away our business.” (Gerstner, 2003: 204-5)

He then went on to criticize the culture inside IBM that tolerated “failure” and IBMers who resisted change. A fundamental change in values and beliefs had to take place for the new practices to become established. In a message to employees in August 1995, he told them “We operate as an entrepreneurial organization with a minimum of bureaucracy and a never-ending focus on productivity” (quoted in Neff, 2012: 40). Of course in an organization that employed tens of thousands of employees, the issue at the heart of matter was not abolishing bureaucracy. Emphasis was rather on operating as an ‘entrepreneurial organization’ that expected commitment of employees to organizational goals without offering any guarantee in return (a model that was commonplace in software companies, see discussion of Microsoft practices below). There is little irony, therefore, in the fact that Gerald Czarnecki, who was hired as head of human resources to make IBM ‘leaner’, was let go within a year reportedly because he fell short of the specified objectives in terms of reduced headcount (see Lazonick, 2009: 86).

---

<sup>12</sup> Uchitelle, L. (1996) ‘More downsized workers return as rentals’, *New York Time*, 8 Dec. 1996.

<sup>13</sup> Lotus had laid off employees before being acquired by IBM. In 1992, for example, it had laid off an unspecified number of 300 employees who were working on its SmartSuite software as part of “a reallocation of resources”. At the time Lotus had 8000 employees of whom 3000 were software developers. Weil, N. (1992) ‘Lotus lays off suite developers’, *InfoWorld*, 10 Nov, 1992.



### 8.3.2 Other computer companies

Because of IBM's prominent position as an industry leader and pioneer of employment security, its fall from grace became a clear indication that the time had come for new norms to replace the old ones. In interviews with nearly a dozen corporate executives, a New York Times reporter found that most of them considered IBM as "a frightening example of a company that had ignored competition for too long and must now engage in emergency job shedding."<sup>14</sup> But IBM was not the only company that had such practices. HP was another company that, like IBM, had provided employment security for a long-time but ultimately experienced a similar transformation. Although HP did not have an official 'employment security' policy, it had effectively avoided layoffs throughout its history. Even after change of policy in IBM, HP managed to avoid layoffs throughout the 1990s. In 1996, HP's manager of human resources was quoted as saying:

"I think [the employability doctrine] is just a rationalization for not being able to provide employment security....We feel very strongly about employment security. We still cherish careers [with the company]" (quoted in McNerny, 1996: 6).

He had also emphasized the benefits of low turnover (around 6.5%) for the company. Nevertheless, HP changed just a short while after Carly Fiorina was brought in as CEO in 1999. In 2001, she pushed through a controversial merger with Compaq and immediately laid off 6,000 workers. This was despite the fact that HP employees had voluntarily taken a 10% cut in their salaries, saving the company \$130 million in three months.<sup>15</sup> Under Fiorina, HP went on to lay off 30% of its employees (about 45,000 individuals) in four years. But since HP had hired new employees in the same period, its total headcount remained virtually the same (see Lazonick, 2009: 91).

While it is unclear how many software workers were affected through these changes, it is clear that employment policies in the IT industry had been completely transformed. No longer could any employee expect to have employment security. At least as far as commitments of employers were concerned, the "Apple deal" had become the standard deal that industry offered workers. Apple<sup>16</sup> itself underwent several rounds of layoffs in the mid-

---

<sup>14</sup> Uchitelle, L. (1993) 'Strong companies are joining trend to eliminate jobs', New York Times, 23 Jul 1993.

<sup>15</sup> Konrad, R. (2001) 'Layoffs may spoil HP workers' allegiance', CNET.com, [http://news.cnet.com/Layoffs-may-spoil-HP-workers-allegiance/2100-1017\\_3-270656.html](http://news.cnet.com/Layoffs-may-spoil-HP-workers-allegiance/2100-1017_3-270656.html) (accessed 10 Mar. 2015)

<sup>16</sup> The only 'new' company that seemed to avoid layoffs consciously was Cisco, which had no layoffs before 2001 even though it made many acquisitions in the mid 1990s. Its number of employees rose from 254 in 1990 to about 44,000 in 2001. However, the dotcom crash forced the company to lay off about 8,000 employees (including part-time workers). Given the range of Cisco products, it is unclear if software workers were affected. Wong, W. (2001) 'Cisco to cut up to 8,000 workers', CNET News, <http://news.cnet.com/cisco-to-cut-up-to-8000-workers/>; Accessed 10 Mar 2015.

1990s (see table 8-2). But in that period, many of Apple's software workers were working in subsidiaries and joint ventures. Besides Applesoft (Apple's main software division), there were two software joint ventures with IBM (Kaleida and Taligent) and a wholly-owned subsidiary called Claris. When each of the joint ventures were discontinued in the mid-1990s, most of the employees were let go (200, for instance, in the case of Taligent).<sup>17</sup> Similarly, when Claris was renamed FileMaker and part of its product line was taken away in 1998, Apple laid off 300 individuals (about 40% of employees).

Year	1981	1985	1990	1991	1993	1997
Total number of employees	1,000	6,000	12,500	15,600	16,000	13,400
Number of laid-off employees	40	1,200	400	1560	2500	2700

Table 8-2: Layoffs at Apple 1980-2001. In 1997, 1400 'contractors' were also laid off.

### 8.3.3 Software companies

Employment relationships at software companies continued the pattern that had been set in the previous decade. Companies typically did not offer employment security and the fate of employees depended on the products with which they were involved and how it performed in the market. Whenever a product seemed to have lost its prospects in the market, it was highly likely that companies would terminate its product line and the related employees. Jobs were further destabilized by waves of merger and acquisition that occurred from time to time. Almost every time a company was acquired or two companies were merged, some employees were laid off. Although technical (including software) workers were not usually affected in successful mergers, in less successful cases they were laid off along with other workers.

Novell is perhaps a good example of this policy. Its move away from hardware into software began with layoffs. Then, in the early 1990s, while dominating the market for network operating systems, it started acquiring a number of software companies in order to become an alternative to Microsoft. For instance, it acquired Digital Research in 1991, Unix System Laboratories (from AT&T) in 1992, Quattro Pro division (from Borland) and WordPerfect (both in 1994). Novell's objective was to become an alternative to Microsoft by gaining independence from MS-DOS (through Digital Research and Unix Laboratories) and expanding its line of office productivity applications (WordPerfect and Quattro Pro).

<sup>17</sup> Ziegler, B. (1995) 'IBM, Apple, H-P to disband Taligent; Big layoffs loom at software venture', *Wall Street Journal*, 1 Dec. 1995.

Therefore, Novell's headcount jumped from 2,729 at the beginning of 1991 to 10,451 in 1993. However, when the strategy seemed unworkable and the company started to make losses in 1995, Novell sold all of these divisions within two years. Several rounds of 'realignments' and layoffs followed, sometime with the intention of making those divisions more attractive to potential buyers (see table 8-3). For instance, when Novell decided to sell WordPerfect, it laid off 420 employees from that division (mainly in marketing and sale).<sup>18</sup> In 2001, it expanded its consultancy services by acquiring Cambridge Technology Partners (with 3400 employees). The acquisition increased the number of Novell employees that provided consultancy to customers from 350 to nearly 3,000.<sup>19</sup> But in the same year, Novell laid off more than 1,400 employees.<sup>20</sup>

Year	1993	1994	1995	1996	1997	1998	1999	2000	2001
Number of employees	10,451	8,457	7,762	5,818	4,797	4,510	5,629	4,893	6,521

Table 8-3: Number of Novell employees between 1993 and 2001.

Microsoft, in contrast, avoided large acquisitions until when it was well established in the industry. Its first major acquisitions were simultaneous with its entry to the Internet industry (e.g. Hotmail and less-known WebTV Networks in 1997). Still most of these companies had a relatively small number of employees (in 1997 Microsoft had more than 22,000 employees worldwide). Therefore, following its acquisitions, Microsoft did not lay off any significant number of employees.<sup>21</sup> In 2001, Microsoft made its largest acquisition (in terms of both value and number of employees) when it acquired Visio Corporation (with 675 employees) but still no employee was laid off.<sup>22</sup> However, there were other mechanisms in place that kept the company lean by periodically dismissing individuals (or small groups). Almost none of these dismissals were ever reported. But since many of these practices became commonplace in the industry, I will describe the hiring and employment policies of Microsoft below.

<sup>18</sup> Just before Novell acquired WordPerfect, WordPerfect had laid off 1,000 employees (or 20% of its workforce).

Horwitt, E. (1994) 'Novell gets go-ahead for buyout', *ComputerWorld*, 16 My 1994.

*Network World* (1995) 'News briefs - Novell Negatives', *Network World*, 20 Nov. 1995.

<sup>19</sup> Anthes, G.H. (2001) 'Inside Novell's one net strategy', *Network World*, 12 Nov. 2001.

<sup>20</sup> *Network World* (2001) 'Layoffs continue at Novell', *Network World*, 19 Nov 2001.

<sup>21</sup> Nevertheless, according to Barr (2001: 143-5), when in 1998 Microsoft sold Softimage (a Montreal-based company that it had acquired in 1994), Softimage employees were prohibited from transferring to Microsoft for one year and their Microsoft stock options were converted to Avid stock options. Since its acquisition by Microsoft in 1994, Softimage had hired 100 employees, reaching a total of 300 employees. See 'Microsoft SoftImage to be Acquired by Avid Technology', <http://news.microsoft.com/1998/06/15/microsoft-softimage-to-be-acquired-by-avid-technology/>, Accessed 10 Mar 2015.

<sup>22</sup> 'VisioCorp Timeline', <http://www.visiocorp.info/timeline.aspx>; Accessed 10 Mar 2015.

### ***8.3.4 Employment policies and decisions in Microsoft***

Before discussing Microsoft's employment policies it is important to present a picture of Microsoft's hiring practices. Microsoft made the majority of its recruitments for programmers and other software workers by employing recent graduates of universities (often directly from the university campus). This tradition of hiring was rooted in early years of Microsoft, when Gates, Allen or other members of senior management personally interviewed university graduates to select from among candidates. These hiring decisions were based on interviews but aptitude tests, which were still common in the late 1970s and early 1980s, were noticeably absent. Instead, programmers were asked some questions that would ostensibly demonstrate their approach to problem solving. According to David Thielen (a former employee):

“One of the old favorites at Microsoft was to ask how many gas stations there are in the United States. The interesting part was not the final answer, but how people went about solving it.” (Thielen, 1999: 29).

Accordingly, the company looks for “people with smarts” (p.25) but “a degree is not required”, because “[i]n software you have numerous brilliant people who have not even graduated from high school, and the industry is used to paying them just as well as people with degrees” (p. 31). Therefore, candidates are expected to demonstrate (on the day of interview) their ability to write small computer programs for specified problems. Similarly, Cusumano and Selby (1995: 98-100) report that Microsoft recruiters look for individuals “who can demonstrate general logical capabilities and work accurately under pressure”. A degree in computer science is preferred but not required. Instead, according to Cusumano and Selby, candidates have to be “expert” C programmers (see also p. 92).<sup>23</sup>

On the other hand, consistent understaffing and working long hours under pressure increase the rate of burnout, especially among developers and testers. Richard Barth, a senior product manager, told Cusumano and Selby that part of this was attributable to Microsoft's “conscious policy to hire about half the number of people we think we need...[This] minimizes bureaucracy. On the other hand, we tend to select people who would burn themselves out anyway” (Cusumano and Selby, 1995: 94, see also Thielen, 1999: 95). Another manager, described the recruiting process as “outstanding in hiring workaholics”, that is people who effectively live in their offices, occasionally working non-stop for three

---

<sup>23</sup> In contrast, many testers have degrees in science. There was one tester for every twenty programmer at Microsoft (compared to IBM's rate of one for every programmer, see Ferguson and Moris, 1994: 70). Adam Barr, a former software developer, has recounted the conflicts that existed between developers and testers in his book (Barr, 2001).

or four days (p.95). This is, of course, under an enormous amount of pressure. A software developer described the situation like this:

“You never get laid back at Microsoft. You always have this tension: Am I doing enough, and have I done everything? And it’s stressful. I know I am not going to die at Microsoft. Another two or three years, and I’m out of here”. (Cusumano and Selby, 1995: 94)

However, according to estimates, the burnout rate for software workers is around 3 percent, which is noticeably lower than the company average (employees, including new hires, leave at a steady rate of ten percent in their first five years with the company). In the early 1990s, Microsoft also established periodical performance reviews in which, based on managers’ assessments, the bottom five percent of its workforce were laid off automatically every six months (Ferguson and Morris, 1994: 176). Nevertheless, Cusumano and Selby (1995: 95) report that, according to Microsoft’s director of development, these layoffs had not affected ‘developers’.

Even if developers were spared from such measures, it is highly likely that they had affected other software workers like program managers, product managers, customer support engineers and user education staff. Barr (2001: 91) recounts that as part of the employee review process, employees submitted reviews of their managers too. It is notable that jobs in Microsoft lacked formal definitions and therefore, there were no formal degree or qualification requirements for any job. According to Cusumano and Selby (1995: 91) this was rooted in the fact that Microsoft did not have clear definitions of these roles and they did not correspond to university curricula. Subsequently, jobholders in those roles were allowed to define their own job descriptions and were actively involved in the hiring of new employees. Given the rate of burnouts and periodical layoffs, there is little stability in most jobs. Instead, new hires are expected to learn by doing with a limited amount of mentoring from more experienced employees. This is a replacement for “investing heavily in training programs, formal rules and procedures, or even detailed product documentation” (Cusumano and Selby, 1995: 105). Nevertheless, in order to retain and reward its technical workers, Microsoft defined and put in place formal career paths, which in the case of developers, usually lead to managerial job (like product manager). These career paths were mainly based on years of experience with the company as well as performance (p.116).

Beside the normal employees, Microsoft treated a significant number of its employees as ‘temps’, even though they worked nearly full-time and only for Microsoft. The number of temporary workers at Microsoft grew from 440 in 1989 (around 10% of employees) to 6,000

in 2000 (15%).<sup>24</sup> These employees usually were responsible for tasks that were not considered to be at the core of Microsoft's business processes and Microsoft did not want to provide them with even the minimum of benefits that it had offered other workers. Therefore, temporary workers were ineligible for vacation or sick leave, company--paid health insurance or pension plan and discounted stock options. While the majority of 'permatemps' were employed in non-technical jobs (such as technical writers or those who prepared content for Microsoft's website), there were programmers and testers among them. To many of these workers, this was simply a "caste system"<sup>25</sup> that discriminated against temporary workers who in terms of skills and experience were not substantially different from normal employees. In 1996, a court of appeals ruled that Microsoft had misclassified its 'independent contractors' and they could be considered 'common-law' (that is, normal) employees for the purposes of eligibility for the discounted stock options. However, it did not require Microsoft to provide other benefits for temporary workers. The long-term impact of the ruling was that Microsoft (and other companies) ensured that temporary workers were indeed periodically laid off so that they could not be considered regular employees or claim similar benefits (Hyde: 2003: 122-3).<sup>26</sup> After a lengthy period of negotiations and court battles, Microsoft finalized a settlement with permatemps in 2005 by agreeing to pay about 93 million dollars to more than 8,500 claimants.<sup>27</sup>

Nevertheless, even normal employees were (and continue to be) expected to 'continually prove their worth' in order to secure their position in the company. Regardless of what an employee has accomplished (for the company) or what the employee is capable of, when that employee is no longer needed, he or she is laid off. In his book on the "secrets of Microsoft management", Thielen describes Microsoft as an "entrepreneurial start-up":

"If employees burn out doing a job, however awesome, they are likely to be cast aside once the project is completed because they are no longer of use to the company...This approach -using up employees and casting them out, albeit with stock options, when they are no longer needed- is clearly ruthless. When taken to extreme, as it is at Microsoft, it is a heartless treatment of workers. However, it is

---

<sup>24</sup> It is unclear whether these employees were considered in the annual headcounts provided by Microsoft. Figures are from Lieber, R (2000) 'The Permatemps Contretemps', Fast Company, no. 37. <http://www.fastcompany.com/40787/permatemps-contretemps> Accessed 10 Mar 2015.

<sup>25</sup> 'Temporary workers: Microsoft case', Kitsap Sun, 13 Feb 2000.

<sup>26</sup> Microsoft, for instance, put in place a mandatory "100-day wait" that every temporary worker at Microsoft had to go through after working at Microsoft for one year.

Andrews, P. (2000) 'Microsoft drops TaxSaver software', Seattle Times, 24 Mar 2000.

<sup>27</sup> See Agreed order approving distribution plan and service provider fees of Vizcaino vs. Microsoft, United States District Court Western Washington Court at Seattle, No. C93-0178C. 30 Sep 2005. Accessible at <http://bs-s.com/pdf/vizcaino/Microsoft-Agreed-Order.pdf>; Accessed 10 Mar 2015.

also an approach that is more profitable than carrying dead wood.” (Thielen, 1999: 98-9, see also p. 76)

At the same time, rigorous understaffing in projects means that it is often likely that when one project is finished, another position is made available. The other position may not be the same as the one that the employee was employed in, but the onus is on the employee to learn and adapt him/herself for the new position. As Thielen explains (p. 100):

“A position may not be the exact job someone wants, but there are opportunities ... Further, people are forced to make a decision to either learn a new skill that matches the company’s existing needs or leave. The alternative is keeping people with obsolete skills, and once again, that’s not just an extra expense but a drag on the company’s productivity.”

Some of Microsoft’s practices, such as recruiting mostly from campuses and establishing career paths were (and remain) relatively uncommon practices.<sup>28</sup> But the broader framework of employment relationship between Microsoft and its employees represent the norm of relationships in the industry. In this sense, it is notable that hiring decisions continue previous practices in some respects, while breaking with them in other respects. Thus, there is still no educational requirement or well-defined basis of skill for entering the software labor market. Entrepreneurs like Gates and others who had entered software work without any qualification were unlikely to impose such limits on hiring of new people. Nevertheless, the primary criteria for hiring continued to be some vaguely defined mental capability that was used in previous episodes as well. But instead of using extensive aptitude tests, at Microsoft at least, emphasis was put on testing employees programming capabilities on the spot.

Moreover, as noted before, “Apple Deal” became the standard mode of work in the industry, including in software work. Employers provided no employment security guarantees and minimal amount of training.<sup>29</sup> Gradually, other companies adopted periodical performance-review-and -layoff programs and ruthless cutting of employees who have ‘lost their worth’. What Ferguson and Morris (1994: 176) recognized as the “Silicon Valley model” became the new norm of dealing with employees. In this model, according to them, “everything is subordinate to performance: rank, tenure, age, status, prestige are fundamentally irrelevant. Compensation is closely tied to profitability or stock performance”. Although some analysts,

---

<sup>28</sup> It is not clear to what extent Microsoft’s approach to the selection interview process has been adopted by the rest of industry. Barr (2001) claims that, as more and more companies are staffed with Microsoft employees, more and more companies see the “Microsoft Way” as the proper way of hiring. What is clear is that the old aptitude tests had lost their credibility.

<sup>29</sup> There are exceptions to training provisions that I discuss in the next section.

including Ferguson, Morris and Saxenian (see Hyde, 2003: 219-222), saw these systems as “the most unbuffered meritocracies imaginable”, they were in fact a new arrangement for linking skills and compensation in which companies adjudicated (unopposed and free of regulation) the skill-worth of individuals as well as how to reward them.

As far as software work is concerned, the imbalance of power had remained more or less at the same level throughout the history of the industry. But while under the traditional norms of employment relationship, companies felt obliged to retain and retrain their employees as long as they could, the new deal at work not only freed companies from any obligation, but also gave them enormous power to define what the required skills were at a much more pragmatic level. ‘Skill sets’ which comprised of mastery of specific tools for specific applications became the norm. As an editor of *InformationWeek* put it in 1997, “the vast majority of companies...treat filling IT jobs like buying PCs, looking to fill a specific spec sheet with the lowest price”.<sup>30</sup> Moreover, many software companies turned to external developers as allied (but third-party or self-employed) sources for expansion of their dominance. For instance, in 2001, while Microsoft had slightly more than 40,000 employees, there were an estimated 5 million developers working on Microsoft’s platform.<sup>31</sup> This was largely made possible through a new institutional arrangement for provision of training that emerged at about the same time that the new norms of employment became dominant. In the next section, I discuss this new arrangement.

#### **8.4 A new institutional arrangement for vendor-provided training**

In the previous section, I analyzed the new employment relationships that became prevalent in the IT industry in this period. In this section I explore parallel changes in the provision of training in the industry. As I will show, these changes led to a new institutional arrangement that was rooted in changes in the competition. There are two sides to these changes: on the one hand, as I mentioned before workers were now responsible for retraining themselves and could not expect employers to provide training. As a report in *Datamation* in August 1990 put it, “Retrain, or die” was the message that employers communicated to their employees.

---

<sup>30</sup> Gallagher, S. (1997) ‘Labor shortage? Look harder’, *InformationWeek*, 1 Dec 1997.

In a survey of changes in skills required in IS job ads between 1970 and 1990, Todd, McKeen and Gallupe (1995: 1) found that “contrary to expectations, the relative frequency and proportion of stated technical knowledge requirements [defined as hardware/software specific skills] in ads have increased dramatically”. A similar pattern was observed by Gallivan, Truex and Kvansy (2004) in a study of “a dataset of job advertisements for IT professionals” between 1988 and 2000 (see also Hyde, 2003: 231-2).

<sup>31</sup> Mundle, C. (2001) ‘The Commercial Software Model’, Prepared text of remarks by Craig Mundle, Microsoft’s senior vice president at New York University Stern School of Business, 3 May 2001. <https://web.archive.org/web/20020201210623/http://www.microsoft.com/presspass/exec/craig/05-03sharesource.asp> Accessed 10 Mar. 2015.



But this does not mean that employers in the IT industry stopped providing training. Rather, they became more focused on providing training externally. A new institutional arrangement, centered on vendor-provided certifications, emerged and immediately became an important part of the regime of skill formation in IT work. In this section I will discuss the emergence of this regime and conclude by discussing the implications of these changes for workers' skills and their work.

I have already mentioned that companies like Microsoft drew on the popularity of programming languages like BASIC to promote their business as well as create new products. On the other hand, workstation vendors used UNIX partly because of its networking capabilities, partly because it could be acquired for free and partly because they were very popular in universities from which most of the target market for workstations (i.e. scientists and engineers) graduated (Ceruzzi, 2003: 281-285). Thus they did not need to make any extra effort to publicize the skills that were required to use their systems. But other companies did not have similar advantages.

For example DEC, which used its proprietary operating system (VMS), seems to have offered very limited (or nothing at all) in terms of training programs. Consequently when its VAX systems became highly popular in the mid 1980s, DEC was unprepared to supply enough skilled workforce<sup>32</sup>. A "hiring war" ensued (on the east coast) in which DEC was competing along with others to hire programmers to ensure the continual growth of its business.<sup>33</sup> While the subsequent fall of DEC is often attributed to problems in its products as well as availability of low-cost powerful PC networks, one may suggest that these problems were augmented by the shortage of skilled programmers. In a letter to *Computerworld*, a Virginia-based practitioner commented that a reason for the shortage of experienced VAX programmers is that

"in Washington D.C., at least, no firm is willing to pay the money to train new people. The general philosophy of hiring in this market is to raid competitors to staff a short-term one-time contract and then start all over again".<sup>34</sup>

While raiding and poaching was not new, arguably, it was the relatively small size of the related labor market and its geographical concentration that had resulted in the visibility of

---

<sup>32</sup> An important cause of the sudden rise in the popularity of DEC VAX series seems to have been its use in the Wall Street. By one account, number of installed VAX computers rose from less than 3000 in 1982 to 43000 in 1987. See Sullivan-Trainor, M, and Babcock, C (1987) 'Mainframe software's new order', *Computerworld*, 8 Sep. 1987.

<sup>33</sup> In 1985 alone, DEC hired 5000 sales and service personnel. See Wilder, C. (1986) 'DEC finishes record year', 4 Aug. 1986. Raimondi, D. (1986) 'VAX boom triggers hiring war', *Computerworld*, 1 Dec 1986.

<sup>34</sup> Kilpatrick, K.P. (1988) 'How to end the VAX hiring wars', *Computerworld*, 22 Dec. 1986.

such practices. In any case, it shows that DEC did not have a proper training program in place.

In contrast, other companies started training programs that were aimed at ensuring that adequate skilled workers were available to support their products. Novell is often considered as the first vendor to offer certifications for workers in support of its products (e.g. Ziob, 2000). In 1989, Novell established its Certified Netware Engineer (CNE) program and, in the first year, trained around 100 workers.<sup>35</sup> By providing these training programs, Novell was hoping to diffuse the pressure of after-sale services that most clients needed. It also established a Netware Service Organizations program that consisted of companies that agreed to offer NetWare support services by employing Certified Netware Engineers.<sup>36</sup> Novell did not typically employ those who were certified. Rather, they were either employed by user organizations (i.e. Novell's clients) or members of the support organizations. They could also work as independent consultants offering services to user organizations.

In essence, these training programs were not different from the earlier forms of vendor-provided training that had started in the 1950s. Given that no academic program of the time was offering relevant skills, these programs provided candidates with enough skills to find employment.<sup>37</sup> What was different, however, was that it was unclear what the value of these certificates was and for whom (Filipcak, 1995). Clients had always preferred evidence of actual experience to certificates.<sup>38</sup> Novell was not offering any guarantee of employment and market for its products were not so secured to offer assurance about the prospects of employment for the newly trained skilled workers. Subsequently, industry analysts and practitioners filled pages of newspapers and trade journals with discussions about the value (or lack thereof) of these certificates<sup>39</sup>. There was even some renewed interest in the issue of

---

<sup>35</sup> Interestingly, according to some reports, original Netware developers like Drew Major were temporary workers at Novell when they developed Netware. See 'Success from failure', [http://www.pbs.org/opb/nerds2.0.1/serving\\_suits/novell.html](http://www.pbs.org/opb/nerds2.0.1/serving_suits/novell.html) Accessed 10 Mar 2015.

<sup>36</sup> By 1990 this included HP, Xerox and Prime. See Buerger, D. (1989) 'Move over Netware?', *InfoWorld*, 18 Jul. 1989.

Wylie, M. (1990) 'Novell retreats further from technical support role', *InfoWorld*, 1 Jan. 1990.

<sup>37</sup> As an indication that this situation was almost mirroring the early years of computer industry, one may note that there was a Netware Users International organization similar to IBM SHARE.

<sup>38</sup> Early Novell exams were pen and paper tests that required workbook study rather than any actual experience with the software. Computer-based and more practical tests were introduced later. Some companies provided opportunities for their employees to earn Novell certificates (costing "several thousand dollars per person") as bonuses. See LaPlante A. (1992) 'LAN/WAN skills sought after in IS shops', *InfoWorld*, 8 Jun. 1992.

<sup>39</sup> Bredin, A. (1992) 'Certification can't hurt – and may help', *Computerworld*, 16 Mar 1992. Rothke, B. (1994) 'Separating CNE hype from reality', *Network World*, 12 Dec 1994. Haber, L. (1995) 'It pays to be sure', *Computerworld*, 22 May 1995. Musthaler, L. (1997) 'Make a good career move: Get certified', *InfoWorld*, 17 Mar 1997. Connolly, P.J. and Yager, T. (2000) 'Do certificates matter', *InfoWorld*, 18 Dec 2000.

licensing of programmers, even though nothing changed following these debates.<sup>40</sup> Views on vendor certificates ranged from those who considered taking these programs as an important step in practitioners' professional career to those who viewed them just as another hype.<sup>41</sup> It was often argued that vendors (such as Novell) are the only group that will benefit from these certificates (both as a source of income and as a source of individuals who will support its products).

Nevertheless, Novell's success as well as popularity of its certificate program suggests that certificates were generally considered valuable by all parties concerned (i.e. vendors, workers and clients). The paucity of CNEs (compared to the expansion of market) created a period of poaching between companies.<sup>42</sup> This was despite the fact that, between 1989 and 1995, Novell trained more than 50,000 CNEs. But since CNEs enjoyed their special position in the labor market, it is safe to assume that they advocated further use of Novell products. Therefore, the more CNEs were trained, the more demand from them was created. A 1996 report by the International Data Corporation called CNEs "an army of evangelists" (IDC, 1996).<sup>43</sup> In the same year, Novell offered recertification tests as well as new certifications such as Master CNE (for advanced level CNEs), Certified Netware Administrator (for entry-level users) and Certified Netware Instructors (for technical trainers).<sup>44</sup> Soon many other vendors followed Novell and started their own certification programs. Other network companies were among the first to join Novell, most notably Banyan and Cisco. IBM and HP offered certifications of their own. By 1996, about sixty companies were offering various types of certifications, many of whom had franchised their training services throughout the world. It was estimated that, between 1991 and 1995, more than 2 million certification tests were administered in the global IT industry (Ziob, 2000).

Microsoft began its developer training and support programs through establishing Microsoft University in 1987 and setting up bulletin boards.<sup>45</sup> With only a modest beginning (5 instructors providing 15 courses) and slow growth, the university was far less effective than Novell's certification program. In 1991, Microsoft started franchising the courses offered by

---

<sup>40</sup> See, for example, Gotternbarn, D. and Webber J.B. (1994) 'Can programmers commit malpractice', *Computerworld*, 29 Jul 1994.

<sup>41</sup> See Rothke, B. (1994) 'Separating CNE hype from reality', *Network World*, 12 Dec. 1994.

<sup>42</sup> Mehling, H. (1994) 'The great trainer robbery', *Computer Reseller News*, 21 Mar 1994.

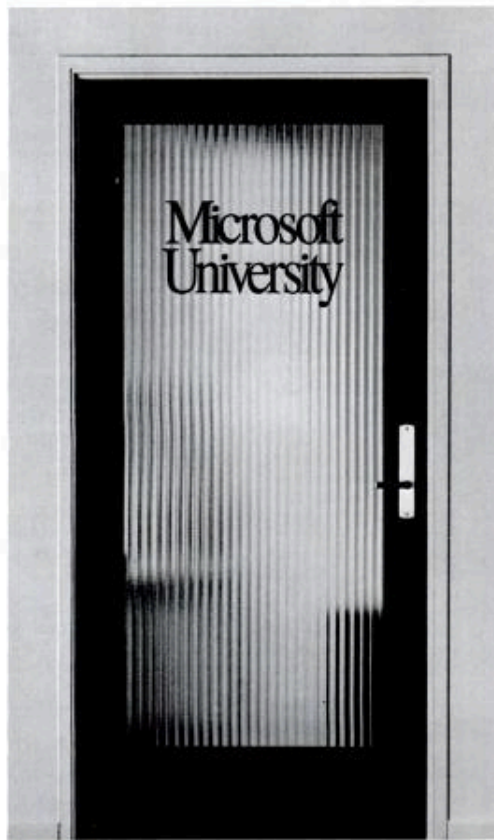
Housman, J. (1994) 'CNE status retains high value in education', *Computer Reseller News*, 30 May 1994.

<sup>43</sup> IDC (1996) *The Fourth Phase of IT Certification*, International Data Corporation, Framingham, Mass.

<sup>44</sup> Tacket, R (1996) 'CNEs cram for recertification exam', *Network World*, 15 Apr 1996.

<sup>45</sup> Bright, D. (1987) 'Microsoft fuses support units', *Computerworld*, 13 Apr 1987.

# We don't have an athletic department, but our graduates get a jump on everyone in their field.



Microsoft University brings the institution of higher learning to an even higher plane. It's a place where programmers and developers learn about Microsoft systems software straight from the source: from instructors with firsthand knowledge of what drives Microsoft systems software.

Our instructors take you through intense, hands-on courses that concentrate on the development of Microsoft® OS/2 and Microsoft Windows applications.

You gain insight and expertise writing software in environments such as MS® OS/2, MS OS/2 LAN Manager and MS OS/2 Presentation Manager. As well as Microsoft Windows/286 and Windows/386.

Courses take place in a lab setting so you not only learn theory, you also gain practical experience by actually writing software. And courses are offered for different levels of expertise.

To receive more information and a copy of the Microsoft University catalog, call 206-882-8080! And if you or your team cannot attend classes at one of our facilities, be sure to ask about our on-site customer training program or video training course that is available.

Either way, we're going to make sure you learn the nuts and bolts of Microsoft systems. So you can make great leaps in your field.

**MicrosoftUniversity<sup>!</sup>**

Figure 8-1: Microsoft University ad, 1988.

Microsoft University to independent training centers but still lacked a certification program.<sup>46</sup> Certification programs began with offering end-user certificates for its popular office products (Microsoft Office User Specialist) but later, in 1992, it targeted computer workers with its Microsoft Certified Professional or MCP program. The program offered several options resulting in different certifications (e.g. Certified Professional, Certified System Engineer, Certified Solution Developer and Certified Trainer). What these training programs offered were skills in SQL Server, programming for Windows APIs and the like.

<sup>46</sup> Mardesich, J. (1991) 'Microsoft extends university to resellers', *InfoWorld*, 14 Jan 1991.

In the same year, Microsoft launched its Microsoft Developer Network (MSDN) to manage its relationship with third-party developers and provide support and assistance to them through software developer kits (SDK) and other programming tools.<sup>47</sup>

After a slow initial start, Microsoft's certification programs grew very rapidly. In the six year of its activity (until 1993), Microsoft University had trained less than 5,000 Certified Professionals.<sup>48</sup> But in 1993, the function of Microsoft University was effectively reduced to producing course materials that were used in 130 authorized training centers that provided Microsoft Certification.<sup>49</sup> By 1994, that number had grown to more than 11,000<sup>50</sup>. Later Microsoft added Internet-related skills to those certifications resulting in specialized certifications such as MCSE+Internet. In the subsequent years, the number of Microsoft Certified Professionals grew rapidly to 100,000 in 1997 and 1,200,000 in 2002.<sup>51</sup> Other software companies were quick to act too. Lotus and WordPerfect provided certifications as early as 1994. Sun began its official certification tests for Java in 1996. Oracle started certification program for its database systems in 1997.

IBM was relatively late in adapting to the new competition environment but eventually switched to the new norm. Simultaneous to the upheaval of employment relationships at IBM, the company changed its training strategy but not towards certification. In 1992, IBM transformed its training services into a new company (called Skill Dynamics) that would provide "competitively priced" training services to both IBM customers and IBM divisions.<sup>52</sup> Building on IBM's history of training programs, Skill Dynamics offered more than 2000 courses through more than 300 training centers<sup>53</sup>. But within a year of inception of Skill Dynamics, internal demand for courses was halved (see below). Externally, IBM's certification program began with the launch of OS/2 version 3.0 in 1994. Certification was the backbone of its marketing campaign for attracting developers, system integrators and consultants to develop products for the new operating system. Eva Rodriguez, senior program manager of the certification program, told reporters that "We want to give industry

---

<sup>47</sup> Johnston, S.J. (1992) 'Microsoft initiates Developer Network service'. *InfoWorld*, 3 Aug 1992.

<sup>48</sup> Haber, L. (1994) 'Microsoft revamps training plan', *Computer Reseller News*, 7 Mar 1994.

<sup>49</sup> Novell, by contrast had almost 700 authorized training systems by 1992. See Raikes, J. (1993) 'Newcomer to sales restructures Microsoft's support', *InfoWorld*, 20 Sep. 1993.

Wallace, P. (1993) 'Client/server training requires top corporate developer training', 8 Nov. 1993. *Computerworld* (1992) 'Certification helps', *Computerworld*, 26 Oct 1992.

<sup>50</sup> Merrill, K. (1994) 'Microsoft training plan builds steam', *Computer Reseller News*, 12 Sep 1994.

<sup>51</sup> Microsoft (2002) 'Microsoft celebrates 10 years of success with Microsoft Certified Professional', <http://news.microsoft.com/2002/04/02/microsoft-celebrates-10-years-of-success-with-microsoft-certified-professionals/>; Accessed 10 March 2015.

<sup>52</sup> Holle, M.S. (1992) 'Full-Service Education and Training Resource; IBM Announces New Education Company: Skill Dynamics', *Business Wire*, 17 Jun 1992.

<sup>53</sup> Sayers, K.W. (1995) 'IBM acquires Catapult', *Business Wire*, 12 Apr 1995.

recognition to the people who work with OS/2....It's patterned after the Novell [Inc.] program because they defined certification in the computer industry.”

Thus, gradually vendor certifications became accepted as an indication of competence. By 2001, it was estimated that there were 630 IT certifications in existence (Bartlett, 2004: 13). With this a new arrangement of skill formation and authentication was institutionalized. Two aspects of these new practices were considerably different from the previous forms of vendor-based training as well as university degrees and certification by associations.<sup>54</sup> First, the new certifications concerned very narrow tasks or specific technologies, which was an outcome of the disintegrated structure of industry. Second, the employment opportunities that these certificates provided were short-lived and uncertain. Therefore, not only the value of ‘certified’ skills depended on their related technologies, but also there was no institutional support for protecting individuals from the fluctuations in the market. This can be contrasted with IBM and others’ long-standing tradition of continuous training of service, sales and support personnel, which took responsibility for retaining and retraining them. In the new structure of competition, companies were not willing to either employ their ‘armies of evangelists’ or make any commitment to them. It was individuals’ responsibility to keep themselves employable by keeping their skills up to date.

Moreover, competitive strategies of companies were extended to the level of individual workers. A significant part of the battle between Microsoft and Novell for control over the network OS was fought in the training sector, with each company trying to provide training to as many potentials as possible, as fast as possible.<sup>55</sup> But perhaps the clearest example of this form of competition can be found in the “Java wars” of the late 1990s, in which Microsoft and Sun competed for controlling Java programming language and its applications. Microsoft tried to undermine Sun’s control by providing software developers with its own APIs and development tools in which Win32 API were integrated. Sun initially tried to block Microsoft’s Win32-oriented specification of Java by offering certification for applications (i.e. 100% Pure Java certificate). But at the height of its battle with Microsoft, in 1999, Sun -joined by AOL, IBM, Novell and Oracle- started a ‘cross-vendor’ certification program for Java developers which aimed at providing “IT professionals” with “the flexibility to develop and validate Java technology skills required for their specific jobs while gaining recognition in the industry”.<sup>56</sup> The “jCert initiative” was subsequently

---

<sup>54</sup> *Certification Magazine* has been published since 1999.

<sup>55</sup> Merrill, K. (1995) ‘Microsoft fuels tech education battle’, *Computer Reseller News*, 24 Apr 1995.

<sup>56</sup> — (1999) ‘IBM, Novell, Oracle, Sun Microsystems and Sun-Netscape Alliance Spearhead Standardized Java TM Technology Certification for Developers’,

endorsed by many other companies and was a key component in promoting Sun's vision for Java.

Thus, institutionalization of vendor-provided certification enmeshed computer workers in the dynamics of market competition. What had been a feature of particular forms of software work became the accepted way of organizing work in the industry as a whole. In this context, it is highly significant that vendor certificates became popular in a period that employment relationships in the American economy were undergoing a major transformation. Laid off workers were looking for ways of returning to work, while those who were not laid off needed some way of ensuring that they remained secure. In this sense, the massive expansion of market for provision of certificates (which included IT companies as well as their franchises) was a result of readiness, on behalf of workers, to embrace such programs. One could argue that these certification programs replaced internal retraining programs of companies without carrying similar employment security obligations.

A significant exception to vendor-based certifications was the A+ Certification program, which was developed by Computing Technology Industry Association (CompTIA). CompTIA started as an association of independent computer sellers and retailers and A+ Certification was designed as a vendor-neutral certification for microcomputer installation and maintenance technicians. In 1994, the first year of its establishment, more than 6,000 technicians passed the exam and became certified.<sup>57</sup> A major source of success of A+ certification was that it had support from virtually all PC manufacturers (including IBM, Compaq, Toshiba) as well as others (like Apple).<sup>58</sup> Later, CompTIA added new areas such as networking and internet technologies to its A+ certification program (Ziob, 2000). While CompTIA initially targeted its certifications for "entry-level technicians", as its reputation grew it expanded its certifications. In 2001, CompTIA introduced Server+ and Linux+, its first certifications targeting system administrators (mainly on Microsoft and Linux platforms). There were also non-IT vendor companies that provided vendor-neutral trainings and certifications (e.g. Prosoft and Learning Tree International).

Part of the appeal of the new training arrangement for the employers was that, not only the trained individuals could act as evangelists (or at least enablers) for IT vendors' products, vendors could earn income from providing those trainings. It is notable that, as a result of changes in the competition and the pressure that all companies faced at one stage or another,

---

<https://web.archive.org/web/20000815085013/http://www.ibm.com/education/certify/news/990517.p.html>; accessed 10 Mar, 2015)

<sup>57</sup> Computer Seller News (1994) 'Testing technicians', *Computer Seller News*, 26 Sep 1994.

<sup>58</sup> Haber, L. (1994) 'A+ certification momentum picks up', *Computer Reseller News*, 16 May 1994.

companies had become more conscious about the cost implications of training programs. This was part of the reason that, just before the arrival of Gerstner, IBM managers came to the conclusion that providing free training for employees without clear indicators for effectiveness of those training programs was wasteful. As Ralph Clark, president of Skill Dynamics, put it, in the past:

“[Employees] had to get a certain number of student a year so we could be sure that they were keeping their skills up to date. ...When you have a system that measures activity and use that as a surrogate for skill building, you get the behavior you measure” (quoted in Geber, 1994: 34).

Clark used the example of an IBM programmer who could learn three or four computer operating systems, at the company’s expense, while he only worked with one operating system, which he needed to know proficiently. While these comments seem to suggest that there were problems with the way in which training courses were provided and measured at IBM, it is clear that these were not the only issue (If so, it would have been enough to develop new measures and practices, cf. Casner-Lotto, 1988; Galagan, 1990). When these changes are considered along with the changes in IBM employment relationships, it becomes evident that at the heart of transformation of IBM training programs was a break with the way in which IBM measured employees’ worth. Because managers were concerned about costs, training activities had to be economically justifiable.

Thus, shortly after being appointed as vice president of human resources, Gerald Czarnecki blamed IBM’s previous management for adopting a wrong ‘management philosophy’:

“The assertion that IBM’s training reinforced the wrong culture may be true....But you can’t blame the training function for mistakes made by management in the running of that company. What you have to conclude is that the management philosophy was wrong.”<sup>59</sup>

According to Czarnecky, it was IBM’s culture that was faulty. (We have already seen that, a few months later, Gerstner targeted IBM culture in a similar way.) In the new institutional arrangement, training departments were also supposed to be ‘profit centers’. Tina Podlodowski, general manager at Microsoft University noted “[o]ur training supports the sale of systems products, but it also needs to stand alone as a viable training company”.<sup>60</sup> This, along with the change in the employment relationship that made individuals responsible for their own training meant that the costs and risks of training were both transferred to individuals. Since certifications carried no employment guarantee (by vendor

---

<sup>59</sup> Allen, F.E. (1993) ‘Executive Education (A Special Report): Payback --- When Things Go Wrong: Executive Education Is Sometimes Just a Waste Of Money; And Sometimes It’s a Lot Worse Than That’, *Wall Street Journal*, 10 Sep 1993.

<sup>60</sup> Crane, K. (1991) ‘Getting your education at vendor ‘universities’’, *Computerworld*, 11 Mar 1990.



or in the labor market), if an individual invested in learning skills that for one reason or another lost value, it was that individual's liability.

It is important to point out that I am not arguing that companies (especially IT vendors) stopped training their employees in the new competition. A parallel development of the late 1980s was establishment of so-called 'corporate universities'. Some of these universities, like Microsoft's, were targeted at external developers but many other provided training for vendor employees too. While there was nothing essentially new about these practices (we have seen that IBM opened its 'graduate school' in 1960), their increased visibility can be interpreted as an indication of ascending role of corporates in defining employable skills (see Alagaraja and Li, 2014).<sup>61</sup> For the purpose of our study, it suffices to note that companies like Sun, Oracle and Cisco provided relatively extensive training facilities for their employees. But these facilities were not generally used for retraining individuals in the way that IBM, among others, did (see, for instance, Meister, 1998; Edgerton, 1999). Therefore, the dynamics of skill formation in these companies were not substantially different from companies like Microsoft that did not provide such employee training services.

In any case, the new institutional arrangements drew on existing practices. Corporate universities were rooted in corporate training programs of the previous decades and the various 'authorized' and 'unauthorized' trade schools that emerged to offer training and certification tests were continuing the line of "EDP schools" that had emerged in the 1960s. Moreover, these new certifications could be mass marketed because IT work continued to have no barrier to entry. Similar to the high school graduates who entered computer work after taking a programming course in EDP schools in the 1960s, high school graduates of the late 1980s and 1990s could enter computer work directly after finishing high school. This was partly because IT companies took a direct interest in establishing their operating systems and programming languages as training material in computer courses at high school level.<sup>62</sup> As a result of these changes, there was no longer any need for aptitude tests as an

---

<sup>61</sup> Another early example, McDonald's Hamburger University, was established in 1963. The changing attitudes towards the role of training activities in corporations can be seen in the business-pitched approach of many of the books that have been published about the topic (e.g. Meister, 1998; Allen, 2007). A more cool-headed assessment is provided by Jarvis (2001).

<sup>62</sup> For example, the Cisco Academy Networking program began in 1997 to train high school students across the United State. By 2002, more than 468,000 students across the world (149 countries) had participated in the program. By that time, it was providing not only Cisco certifications but also basic IT certifications and courses sponsored by Sun, HP, Adobe and others. Carless, J. (2005). 'Cisco networking academy program: Five-year fast facts', [https://web.archive.org/web/20081022213608/http://newsroom.cisco.com/dlls/hd\\_040303.html](https://web.archive.org/web/20081022213608/http://newsroom.cisco.com/dlls/hd_040303.html) (accessed 10 Mar. 2015).

indication of programming or any other computer-related abilities. While the debate over aptitude tests was alive as late as 1982,<sup>63</sup> their significance and practical relevance seem to have declined steeply since mid-1980s.

I have argued in this section that the institutional arrangements surrounding vendor-based provision of training changed in this period. These changes were partly due to the changing structure of industry and partly due to the wider concurrent changes in employment relationships. As a result of these changes individuals had much less institutional support for protecting their skills and their employment relationship.

### **8.5 Labor market**

With the exception of a slowdown in 1985, the computer industry continued to expand and grow for the rest of the 1980s and through the 1990s (Einstein and Franklin, 1986; Shank and Getz, 1986). IT labor market too continued to grow (Hatch and Clinton, 2000). However, due to intensification of competition in the computer manufacturing, most of the new jobs were created in the software and services sector (Plunkert, 1990). Changes in the competition (which affected many companies) in parallel with the changes in employment relationship and massive restructuring programs of the early 1990s, meant that many IT workers were forced to change their jobs. The new certification programs provided a major basis for people with limited or no experience to enter the labor market. The number of IT workers soared as people (laid off employees as well as those with limited or no experience) rushed to take advantage of the opportunities created by the Y2K problem and the surge in website development. Therefore, the schismatic and differentiated structure of labor market was shaken up in this period, leading to an even more fragmented structure.

At the same time, while older associations continued to operate in their niche-like spheres, new 'professional' and worker associations emerged in software work. The failure of these associations to join forces and form a united body meant that they were incapable of taking action in response to any of these developments and effectively irrelevant to developments in the labor market. The best they could do was to offer training and certifications, which ultimately had to compete with the much better organized vendor and industry-sponsored certifications. In this section, I first consider the role of worker associations and employee groups in this period. Then I will consider some of the lobbying efforts that IT companies made and their consequences for IT workers. I conclude by briefly discussing the situation

---

<sup>63</sup> Karten, H. (1982) 'Pros and cons of standardized aptitude tests for programmers', *Computerworld*, 12 Jul 1982. Tharington, J. (1982) 'From within', *Computerworld*, 8 Mar 1982.

in the aftermath of dotcom crash, during which the absolute number of IT workers (including software workers) fell dramatically for the first time in the history of IT industry.

### ***8.5.1 Worker associations and employee groups***

Older associations that were formed in the previous decades continued to operate into the new era but they even struggled to define their relevance in the emerging market for skill provision. ICCP, the main association created for the purpose of providing certifications, had less than 50,000 certificates issued. This was despite the fact that the number of its constituent associations had grown to 21.<sup>64</sup> ACM and IEEE Computer Society, which had become established as primarily ‘scientific’ and educational societies, had limited role beyond shaping academic research and advising higher education programs and lacked any independent power to influence the labor market. Other associations were either contented in their niche or had disappeared altogether. Many workers, for one reason or another, felt that they were not served by older associations and formed new associations. But these associations too were oriented towards creating a technical community for a limited a group of practitioners. In this sense, they were very similar to most of the associations that were formed before the 1980s. None of these associations tried to exert control over the labor market or represent the interests of their members by either engaging directly with employers or lobbying the government. Their primary objectives were entirely ‘apolitical’ in the sense that avoided alignment with the interests of any group of employees or workers in relation to either employers or the government.

For instance, the Network Professionals Association was originally formed as the ‘CNE Professional Association’ in 1990 as a means of communication between those who had passed Novell’s certification exam. In 1994, the association decided to change its name and become a vendor-neutral association for all network workers. It defined its mission as advancing “the network computing profession” and its objective as uniting “network computing professionals” in a worldwide association, determining their needs and interests and delivering “programs and services to meet those needs and interests”.<sup>65</sup> However, it showed no interest in regulating the labor market for network professionals and limited its activities (to a large extent) to providing training and skills through courses and conferences. In 1996, it started to provide its own brand of certification, dubbed Certified Network Professional (CNP). By the end of decade, its 7,000 members in the United States was roughly equal to 7% of the number of people who worked in networking-related jobs in the IT industry alone (Benner, 2002; see table 8-6).

---

<sup>64</sup> Haight, M.J. (1993) ‘Cast your vote on certification’, *ComputerWorld*, 22 Mar, 1993.

<sup>65</sup> NPA (1996) ‘NPA overview’, <https://web.archive.org/web/19961112130840/http://www.npa.org/npa/overview.html>; Accessed 10 Mar 2015.

Another association was the International Webmasters Association (founded in 1996) which had 12,000 members across the globe by the year 2000 (Benner, 2002). The stated goals and objectives of this association included providing “superior professional development programs to include a Certified and Senior member designation program for all Web professionals”, providing its members with networking and job opportunities and maintaining “universal standards on ethical and professional practices”. In no way the association tried to regulate the related labor market or indeed its own membership. It invited “[a]ny individual who will contribute to the development, integrity, and the advancement of the Webmaster profession and the objectives of the IWA” to join.<sup>66</sup>

A more or less similar group of worker associations that emerged in the 1990s consider themselves as ‘guilds’ rather than ‘professional associations’. For instance, Benner (2003) describes eight such guilds that have been formed within the Silicon Valley. The main function of these guilds is to “strengthen their members’ employment opportunities through more informal channels of social networking”, help their members “to improve their skills” and improve their members’ “negotiating positions in the labor market”. This latter role is often played by “providing information to help their members advocate for themselves *individually* in the labor market” (my emphasis). Only occasionally these group engage in advocacy (much less lobbying) campaigns.

While insisting that these associations provide important supportive roles for their members, Benner admits that this sort of “occupational associations” provide a “much less tangible” institutional basis than traditional professions and unions, and importantly for our purposes, often fail to influence employers. It is unclear to what extent these associations may contribute to institutionalization of jobs, given that they often see their role as helping their member cope with changes. In my view, these associations are following the same trend that earlier computer associations followed. The fact that three of the eight associations that Benner names are web-related (namely, the “HTML Writers Guild”, the “Silicon Valley Web Guild” and the “Silicon Valley Webgrlls”) but, despite their close proximity, are not related to each other can be interpreted as indicating a highly fragmented labor market, within supposedly “dense social networks”.

In contrast, new forms of collective action began to emerge in the form of collectives of dissenting or laid-off employees of specific companies. One of the first such collectivities

---

<sup>66</sup> ‘About the IWA’, <https://web.archive.org/web/19981203070824/http://www.iwanet.org/about/>; Accessed 10 Mar 2015

was “Employees for One Apple” which was formed after Apple layoffs in 1991.<sup>67</sup> “Employees for One Apple” was a group of around 500 skilled white-collar employees (‘professional workers’) who were protesting against Apple managers’ short-term outlook. The company had recently announced that it wanted to lay 1,500 employees off, just after hiring 1,000 new employees in the previous 8 months.<sup>68</sup> However, it seems that members of “Employees for One Apple” (most of whom were not being laid off) did little or nothing beyond staging a rally in front of Apple headquarters. While it is unclear if they achieved anything, in the subsequent days senior Apple managers agreed to take a 5-15% cut in their salaries<sup>69</sup>. I could find no other activity (or trace) of this group after the events in 1991.

Another collective of employees was the Washington Alliance of Technology Workers (or WashTech), which was formed in 1998 in the aftermath of temporary workers’ class-suit action against Microsoft (discussed in section 8.3). A catalyst for its formation was a development that directly affected software workers. In 1997, a major trade association of IT companies in Washington state (called the Washington Software and Digital Media Alliance), whose members included Microsoft, lobbied the Washington State Department of Labor and Industries to authorize elimination of overtime pay for computer workers who earned more than a certain level. More than 700 individual workers tried to stop this from happening by writing to the department in protest. However, department approved the proposal, demonstrating that workers’ individual voices were ineffective (van Jaarsveld, 2004). Composed primarily of Microsoft’s temporary workers, WashTech became affiliated with the Communication Workers of America, the largest labor union for communications and media workers in the United States. Despite its small size (260 dues-paying members in 2000)<sup>70</sup> and its failure to achieve official recognition by Microsoft or any other employer, it has been influential in introducing employment-related bills in the Washington state in favor of employees through various forms of political action. Although almost none of the bills were passed, they helped draw attention to these issues at public level and limited the manoeuvres that were available to workers.

WashTech has also pressured some Washington state employers to alter some of their employment decisions. For example, this collective added pressure to the then ongoing legal case of temporary workers which forced Microsoft to change the contract and status of many

---

<sup>67</sup> The timing, and perhaps, fate of this group must be considered in the context of a parallel and much more high-profile collective action which was staged by janitors who worked for Apple and a number of other Silicon Valley companies (through intermediaries). See Rudy (2004).

<sup>68</sup> Zachary, G.P. (1991) ‘Apple workers mull collective bargaining push’, Wall Street Journal, 19 Jun 1991.

<sup>69</sup> Lawlor, J. (1991) ‘Troubles bring union talk’, USA Today, 18 Jul 1991.

<sup>70</sup> Andrews, P. (2000) ‘Microsoft drops TaxSaver software’, Seattle Times, 24 Mar 2000.

permatemps into full-time employees between 1999 and 2000 (Wilson and Blain, 2001). But they were not always successful. In 1991, a group of 18 permatemp working (mainly as programmers) in a Microsoft product line called TaxSaver formed a group to address differences between their pay and benefits despite having similar skills and tasks. With the help of WashTech, the group was initially successful in improving the condition of its members. However, shortly afterwards, Microsoft killed the product line and sold the unit with its 100 employees (including 50 permatemps). WashTech also supported workers by providing training, including courses in Flash and HTML. Thanks to a Communication Workers of America agreement with Cisco, WashTech was also able to provide training courses for Cisco certification (van Jaarsveld, 2004).

Another notable employee group is Alliance@IBM, which was formed in 1999 to protest to IBM's decision to change the pension plan of many of its employees to cash-balance.<sup>71</sup> Effects of the proposed cash-balance conversion formula was different for different age groups and some (particularly mid-career) employees were close to losing 30-50% of their expected pensions. This was amid some industry-wide suspicions that employers were systematically discriminating against elder workers (see Hyde, 2003: 226-9). IBM's justification of the plan was that "competition in our industry for skilled, talented employees has never been more fierce than it is today". Gerstner also argued that IBM's old pension plan "was created at a time when employees joined IBM for life" and since this could no longer be the case it had to change.

Some IBM workers took the case to the court (Cooper vs. IBM) but many rallied to form Alliance@IBM. Because IBM was the largest employer in Vermont, Vermont's only senator Bernard Sanders and a few others became directly involved. Just before a hearing was about to take place at the Senate<sup>72</sup>, Sanders convinced Gerstner to alter the formula and provide many more employees the option to choose on their existing plans. After a long judicial process, the U.S. Supreme Court ruled in favor of the employees and IBM was forced to pay according to a settlement reached in 2004. By that time Alliance@IBM had more than 5,000 members and was affiliated with the Communication Workers of America. While Alliance@IBM has continued to raise issues about the conditions of IBM U.S. Employees, it has had limited impact on IBM's decisions or policies since. IBM put in place new pension plans for new hires and gradually replaced the traditional pension plan with the new one.

---

<sup>71</sup> IBM's massive layoffs in the early 1990s seem to have sparked no collective action by workers (and certainly no worker organization).

<sup>72</sup> Oppel, Jr., R.L. (1999) 'IBM does an about-face on pensions', New York Times, 18 Sep 1999.

The employee groups discussed here were in many ways unprecedented in IT work and indeed shocking for some observers at the time. Despite their small size, each had managed to achieve some degree of success in one aspect or another. But they have not led to any large-scale movement towards worker organization (of any form). Their limited impact beyond their immediate environment reflects both the highly fragmented structure of IT work and the established power-imbalance that exists between workers and employers.

### ***8.5.2 The effects of IT companies' lobbying on the IT labor market***

In contrast to worker organizations, which had only limited impact on particular decisions, IT companies (like IBM and Microsoft) successfully lobbied with the government in ways that not only benefited their interests, but also shaped to shape employment relationships at a much larger scale. Moreover, despite their rivalries in the competition, they were happy to cooperate in trade associations as a united group in order to pursue their collective interests. Thus, unlike worker and professional associations, business associations became extremely powerful in shaping IT work.

I have already mentioned the role of Microsoft and Washington Software and Digital Media Alliance in shaping the working conditions of software workers (including permatemps) in the state of Washington. A far more influential role in shaping the working conditions of programmers and other software workers was played by IBM at the national level. The status of computer workers regarding Fair Labor Standards Act (FLSA) had remained ambiguous throughout the previous episode. The issue was finally settled as part of a deal that IBM made with the Congress, even though IBM's original lobbying was unrelated to computer workers. In 1986, at the same time that IBM was at pains to retain and retrain its own employees, it was seeking tax break for its overseas operations. However, budgeting procedures required that the loss of government revenue from taxes be offset by income from other sources. Congress estimated that, since self-employed individuals were more likely to cheat on their taxes than company employees, if self-employed programmers were classified as employees, enough tax revenue would be raised to pay for IBM's tax break. Therefore, in order to provide IBM with the tax break, Congress made two other changes in the law that were aimed at making it more difficult for programmers to be self-employed.

First, the Revenue Act of 1978 was amended to exclude programmers from some simplified rules for claiming self-employment status. The 1986 amendment added programmers and systems analysts to the list of occupations that were excluded from the provisions of that act. But since more programmers and systems analysts had to be employed by companies as a result of the act and their status regarding FLSA had remained unclear, it was likely that

companies ended up paying those employees more. Therefore, Congress made a simultaneous change in the Fair Labor Standards Act to make “computer professionals” (including “computer systems analyst, computer programmer, software engineer, or other similarly skilled worker”) explicitly exempt from overtime payment and the like (see Hyde, 2003: 121-2).<sup>73</sup>

Moreover, unlike certain other categories of employees which, under FLSA, were considered exempt purely on the basis of attaining specific degrees or qualifications, the regulation regarding “computer professionals” specifically held that “[w]hile such employees commonly have a bachelor’s or higher degree, no particular academic degree is required for this exemption.”

Instead, individuals could be considered as exempt when their “primary duties” included:

- (1) The application of systems analysis techniques and procedures, including consulting with users, to determine hardware, software or system functional specifications;
- (2) The design, development, documentation, analysis, creation, testing or modification of computer systems or programs, including prototypes, based on and related to user or system design specifications;
- (3) The design, documentation, testing, creation or modification of computer programs related to machine operating systems; or
- (4) A combination of the aforementioned duties, the performance of which requires the same level of skills.

It is notable that under these provisions, a significant number of software workers (like technicians and those in customer support) are not exempt. Moreover, in the legal tradition, job titles or even job descriptions are considered inadequate for the purpose of judging employees in exempt/non-exempt categories. If employees think that they are misclassified, they have to demonstrate that their daily activities fall within the boundaries defined above. I could find no indication of any negative reaction by employees to the passage of this law or any surge in misclassification claims.

---

<sup>73</sup> Contrary to common perceptions about the boom in self-employment around the turn of century, at 6.4 percent, self-employment in the U.S. economy in the year 2000 was at its historic *low* (and even lower at Silicon Valley, at around 6.2 percent). The percentage of workforce in self-employment had continuously fallen from an estimated 25% at the beginning of the twentieth century to less than 7 percent in the 1970s and had remained at that level for three decades (Hyde, 2004: 112-7).



But perhaps the most important role played by IT companies in this period was their active involvement in raising the cap of immigration of ‘highly-skilled’ workers in the late 1990s.<sup>74</sup> As early as 1952, the Immigration and Nationality Act had defined an H-1 ‘nonimmigrant’ as:

“an alien having a residence in a foreign country which he has no intention of abandoning (i) who is of distinguished merit and ability and who is coming temporarily to the United States to perform temporary services of an exceptional nature requiring such merit and ability.” (Quoted in Lowell, 2000)

The word “temporary” before “services” was removed in 1970, to allow temporary immigrants to take up “permanent” positions. According to Lowell (2000), Congress never defined the criteria for “distinguished merit and ability”. An administrative decision, by the Immigration and Naturalization Service, determined that those applying for such a visa are only required to have attained an undergraduate degree in a related field as a basis for entering the respective field of work.

Nonetheless, the actual screening of visa applicants remained problematic and a dramatic increase in the number of H-1 visa-holders in the mid-1980s led to protests from labor organizations. After conducting a review, immigration laws were reformed in 1990 and attainment of an undergraduate degree as a minimum requirement was inscribed into the law. Moreover, an annual cap of 65,000 was set for H1-B<sup>75</sup> visa holders for the first time (Usdansky and Espeshade, 2000) but they were now allowed to stay one year more than before (i.e. for 6 years) and to apply for permanent residence while in the U.S. Prospective employers were also required to file wages and worker conditions with the department of Labor. For five years between 1992 and 1996, the cap of 65,000 was not reached.

In 1995, the then Labor Secretary, Robert Reich sought to reform the immigration legislation. The proposals for reformation included measures that would decrease the number of H1-B workers and increase the costs of their employment by requiring employers to pay H1-B visa-holders more than the average of comparable U.S. Workers (Usdansky and Espeshade, 2000). Defending the proposals, Reich said:

---

<sup>74</sup> While oversees software activities have had a long history in the American IT industry, offshoring of software became a visible phenomenon only in late 1990s. On the other hand, Indian programmers entered the American IT labor market as early as 1974 (see Aspray, Mayadas and Vardi, 2006). Since the issue of immigrating workers is more directly linked to the issue of skills (as opposed to offshoring which is often justified in terms of economic savings), I will only consider this issue in my research.

<sup>75</sup> H1-A visa category was created in 1989 to separate requirements for nurses from other occupations.

"We have seen numerous instances in which American businesses have brought in foreign skilled workers after having laid off skilled American workers, simply because they can get the foreign workers more cheaply...[The current immigration system] has become a major means of circumventing the costs of paying skilled American workers or the costs of training them."<sup>76</sup>

While no professional association was present at the congressional hearings, a former IBM programmer called Lawrence Richards attended the hearings as executive director of Software Professionals' Political Action Committee, a relatively unknown group formed in 1994. Members of the Austin, Texas-based group were programmers who were "intent on changing immigration laws that were affecting their jobs"<sup>77</sup>. Richards told the congress that labor shortage in IT work was not significant and foreign programmers' pay was about two-thirds that of the U.S. Programmers.<sup>78</sup> Other programmers attending the hearing were present as individuals but shared similar concerns. Collectively, they predicted that lowering of wages would further accelerate the offshoring of jobs, discourage American students from studying for these occupations and ultimately create the very shortage that it is ostensibly designed to prevent.<sup>79</sup>

IT businesses were vocal in their opposition to the reforms. Microsoft, Sun and Intel opposed any further restriction of immigration workers. Harris Miller, president of the Information Technology Association of America, told the congress that

"It appears the provisions of the [new] bill are based on emotion not facts. ...Skilled foreign workers are needed to penetrate foreign markets, to fill vacancies in certain highly skilled positions and to deal with periodic labor shortages".<sup>80</sup>

In an opinion piece in the *Computerworld* magazine earlier that year, Miller had called the proposed legislation 'xenophobic' and argued:

"The U.S. economy needs innovations not ceilings that inhibit entry of innovators. As a nation we must embrace those who create innovation, regardless of their country of birth. To do otherwise, would only deny the creation of wealth and high-paying jobs in the U.S. ...If our country adopts a protectionist immigration policy

---

<sup>76</sup> Branigin, W. (1995) 'White collar visas: Back door for cheap labor?', *Washington Post*, 21 Oct. 1995.

<sup>77</sup> SoftPac (1996) 'SoftPac Information', <https://web.archive.org/web/19961025044910/http://aea.org/softpac/softpac-info.txt>; Accessed 10 Mar 2015.

<sup>78</sup> Betts, M. (1995) 'Programmers to Senate: 'Preserve American dream'', *Computerworld*, 4 Dec 1995.

<sup>79</sup> Branigin, W. (1995) 'White collar visas: Back door for cheap labor?', *Washington Post*, 21 Oct. 1995.

<sup>80</sup> Betts, M. (1995) 'Programmers to Senate: 'Preserve American dream'', *Computerworld*, 4 Dec 1995.

on skilled immigrants in the midst of a global economy, the American information technology industry and American workers will all be losers.”<sup>81</sup>

Other IT and software-related business associations too lobbied the congress, making similar arguments.

Ultimately, IT companies and their allies succeeded in removing such proposals almost entirely from the Immigrant Reform and Immigrant Responsibility Act of 1996, which focused exclusively on illegal immigrants (Usdansky and Espeshade, 2000). By 1998, the table had turned. In 1997 and 1998, the cap was reached and IT industry, as the major employer of H-1B visa-holders, began to raise concerns. In early 1998, Miller cited a “severe shortage of competent and skilled Information technology workers” to argue that:

“The shortage threatens not only the information technology industry, but the growth of the entire U.S. economy, our global competitiveness and the wage stability that is the bedrock of this country's low inflation.”<sup>82</sup>

Later that year, he attended a hearing to lend support to the argument that “Congress should raise or eliminate the cap”.<sup>83</sup> Representatives from Microsoft and Sun reiterated their position regarding their demand for foreign-born workers to fill their open position and to tailor products to specific markets.<sup>84</sup>

President of IEEE, John Reinert, was the only representative from a professional association that attended the hearings. He argued against the suggestion that a major shortage of IT workers is imminent and pointed out that, historically, the labor market for IT workers had been flexible with regards to entry requirements. He further argued that the H1-B program needs to be reformed, but not in terms of the cap but in terms of the criteria used for evaluating applications, attestations received from companies and the investigatory powers of the overseeing agency. He also recommended that companies who apply for hiring H1-B visa-holders should be required to provide evidence that they had not laid off anyone in similar jobs before application.<sup>85</sup>

---

<sup>81</sup> Miller, H.N. (1995) ‘Don’t close the door on immigrant programmers’, *Computerworld*, 10 Jul 1995.

<sup>82</sup> Pear, R. (1998) ‘Higher quota urged for immigrant technology workers’, *New York Times*, 23 Feb 1998.

<sup>83</sup> ITAA (1998) ‘ITAA calls on congress to raise cap for temporary workers’, ITAA press release accessible at <http://shusterman.com/itaatemporaryworkers.html>; Access 10 Mar 2015.

<sup>84</sup> Ouellette, T. (1998) ‘Proposal could pop cap off high-tech visa limits’, *Computerworld*, 2 Mar 1998.

<sup>85</sup> Reinert (1998) ‘Testimony on High Tech Worker Shortages and Immigration Policy’, <http://www.ieeeusa.org/policy/policy/1998/98feb25.html>; Accessed 10 Mar 2015.

Clinton administration initially rejected the proposals to remove the cap. But under pressure, it agreed to discuss the issue with the congress. Ultimately, they reached a ‘compromise’ in which the gap was pushed beyond what IT companies had hoped for in their original proposals (115,000 instead of 80,000-100,000) but in return, they had to provide funds for training and education (in the amount of \$500 per application). The compromise also specified that under certain circumstances, companies had to layoff attestations. But the specified circumstances were only applicable to companies in which H1-B visa holders constituted a high proportion of employees (Usdansky and Espeshade, 2000). One item of the agreement that was subsequently overturned was that the increased cap was only valid for the years 1999 and 2000 and then shall be reduced to 107,000 in 2001 and 65,000 in 2002. When the quota was reached amid the frenzy of Internet activity at the turn of millennium, the cap was increased to an unprecedented 195,000 in 2001, which remained in force until 2003. But this quota was never reached and as dotcom bubble burst the number of applications fell. In 2004, the cap was returned to 65,000 (table 8-5).

Year	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003
Cap	65	65	65	65	65	65	65	115	115	195	195	195
Issue	48.6	61.6	60.3	54.2	55.1	65	65	115	115	163	79.1	78

Table 8-5: Level of cap and number of H1-B visas issued between 1992 and 2003 (in thousands)

(Source: U.S. Department of Homeland Security)

### 8.6 Implications for software work

Thus, this brief episode of political action in the final years of twentieth century illustrated all the problematic characteristics of IT work. The undefined nature of IT skills, which was amplified by the new and more uncertain competition, was at the core of discussions about allowing immigrants into the country. Arguably, this ambiguity was the vehicle that enabled IT companies to operate the system at their will. The new employment relationships and training arrangements meant that companies were unwilling to take responsibility about the reskilling of their own employees but were happy to contribute a small charge towards training programs. Indeed, when it came to helping public and other forms of non-business motivated training programs, some companies and businessmen were particularly boastful of their ‘philanthropic’ or ‘patriotic’ actions and grants, and their ‘commitment’ to further development of American workforce.

At the same time, the complete absence of professional organizations from the debates around the issue of immigrant workers demonstrates the extent to which they had become

irrelevant to the labor market. Indeed, it was in the absence of almost any action by professional associations that some new worker associations like Programmers Guild emerged to challenge the industry’s proposal. While not a labor union, in contrast to the purely technical nature of IT professional associations like ACM, mission statement of the guild in 1998 put “lobbying” at the top of its activities while also mentioning “professional standards”, certification and job placement in the list of its services.<sup>86</sup> Even though the guild played only a minor part in the 1998 debates (by publicly voicing its concern to the newspapers)<sup>87</sup>, it became increasingly active in the subsequent years. Still, with a membership of around 1,000 in 2001,<sup>88</sup> it was too small to have any real impact on the broader labor market or national-level policies.

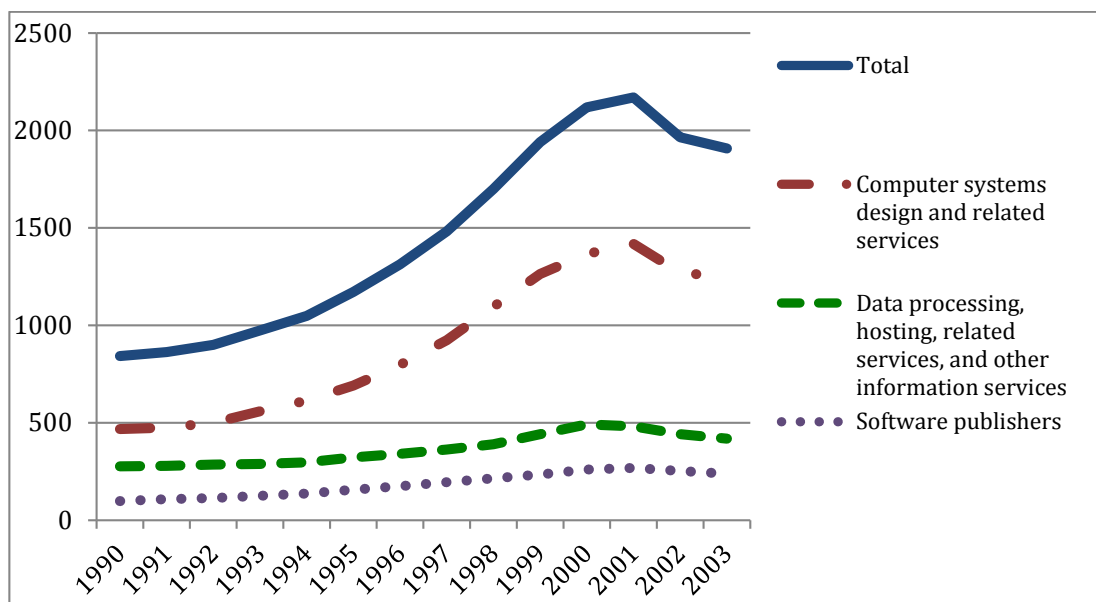


Figure 8-2: Employment in U.S. IT industry, 1990-2003

In the previous chapter, I discussed the varieties of software work that existed in the period up to 1985. As I explained, in that period, each type of software worker was employed in a different part of the industry and under different employment relationships. Animosity between the two sides of the divide some times reached the pages of industry’s trade journals. Cheryl Currid, a Novell-authority-turned-downsizing-advisor,<sup>89</sup> suggested that IS

<sup>86</sup> Programmers Guild (1998) ‘Welcome’, <https://web.archive.org/web/19990508052518/http://www.programmersguild.org/american.htm>; Accessed 10 Mar 2015.

<sup>87</sup> Wayne, L. (2001) ‘Workers, and bosses, in a visa maze’, New York Times, 29 Apr 2001.

<sup>88</sup> Levinson, M (2001) ‘IT guild: A once and future union?’, CNN Online Edition, <http://edition.cnn.com/2001/CAREER/trends/05/09/itworkers.union.idg/>; Accessed 10 Mar 2015.

<sup>89</sup> I am using downsizing here in the more special sense of moving from mainframe systems to client/server computer networks. In any case, a year after this incident she published a book on reengineering organizations.

managers should “plan now to prevent career deadend for Cobol programmers”. She compared Cobol to Latin and argued:

“IS managers should encourage colleges and universities to stop teaching Cobol in schools. It is a waste of time, and many students will have to unlearn what they have been taught.”<sup>90</sup>

Even though her list of suggestions included retraining and reassigning Cobol programmers, she suggested that those who resisted these changes should be fired. The response from some Cobol programmers was such that, in her next article on the topic, Currid said she “expected to read the headline “Cobol coders call for Currid’s head””.<sup>91</sup> She said she had also received support from “fellow anti-Cobol activists”, “who cheered the notion of getting rid of the language”. While reiterating her position, she reflected some of the points that readers had raised, including the slow rate of response to change in universities and the suggestion that “individual IS professional” is the only person who shall assume responsibility for his/her education. The passion and intensity with which these arguments and counterarguments were made show that far from a merely technical object, programming languages like Cobol, had come to define identities, careers and life-investments.

Nonetheless, both sides of the argument were involved in software development and in the period discussed in this chapter, worked under similar employment conditions. This could have brought a degree of cohesion and shared experience among the workers. But the emergence of new areas of work, particularly around networking and website development, brought new sources of variety into software work. While most of the networking technologies that were subsequently used were in existence before the 1980s, when networks emerged as a new layer in the platform of IT work, many programmers’ found themselves lacking in training or experience. Moreover, the proprietary status of networking operating systems (as opposed to shared standard status of Cobol) and the fierce competition that ensued between Novell and Microsoft meant that part of the competition was being fought in the training and certification of IT workers. Not only vitality and utility of skills that networking practitioners learned depended on the vested interests behind the technology, networking practitioners had themselves become part of the vested interest behind technology by investing their time and capabilities. Therefore, whichever side was

---

<sup>90</sup> Currid, C. (1993) ‘Plan now to prevent career dead-end for Cobol programmers’, *InfoWorld*, 8 Mar 1993.

<sup>91</sup> Currid, C. (1993) ‘Cobol has its passionate defenders, but they are a minority’, *InfoWorld*, 24 May 1993.

destined to win the competition<sup>92</sup> one group of workers was certain to lose. It is important to note that this liability and mutual dependence was only meaningful because they were working under a new employment relationship that had shifted most of the risks and costs of learning to individual workers and there was no other form of institutional support (from co-workers or the government) that could protect these workers.

The issue of website development is more complicated because the number of interested parties was more and the range of tasks that could be placed under the umbrella of website development were more varied. We saw that part of the competition surrounding control over the web was fought in attempts to define the ‘essence’ of Java programming technology as well as distributing corresponding skills among workers. But more broadly, web development included tasks that went far beyond programming, including information architecture, content production, graphic and interaction design, maintenance, etc. (see Damarin, 2006). There was no agreement about how these tasks should be organized and assigned to different workers, nor there was any agreement about the skills that were required for performing these tasks.<sup>93</sup> Just as the first generation of PC software developers and entrepreneurs had found themselves re-inventing the basic tools of trade, workers that joined the booming networking and Internet industry found that skills and practices of software development irrelevant or inadequate for their work responsibilities. What they shared, however, with PC software developers was a shared disinclination to work in (or according to) traditional corporate environment, and a willingness to work for unusually long hours under pressure in the hope of deferred rewards in the form of stock options (Ross, 2003; Neff, 2011).

Thus, given the absence of any control over the entry to the IT labor market, lack of a common understanding about the required skills, and employers’ eagerness to hire anyone who could show certified skills or experience, it is hardly surprising that web developers took the many forms and shapes that they took (Christopherson, 2004; see also Stark and Girard, 2009). An early study of these ‘new media workers’ found that they are “largely self-taught” and “spend, on average, 13.5 unpaid hours per week in obtaining new skills” (Batt, et al., 2001: 1). But once the dotcom bubble burst, workers in this area found themselves struggling for survival. Under the new employment relationship, workers were left to their own tools and, even though their skills were by no means outdated, if they wanted to keep themselves ‘employable’, they had to continually reinvent themselves by

---

<sup>92</sup> I am assuming, of course, that given the importance of network externalities of network technologies, it was unlikely that the industry would become divided between two different networking operating systems.

<sup>93</sup> Alexander, S. (1998) ‘Web certifications abound’, *InfoWorld*, 13 Apr 1998.

recombining their skills with new ones, or at least fashion them in new ways. Indeed, in their multiple attempts to get back to work, it was quite typical for laid-off workers to submit different versions of their resume (see Lane, 2011, 74-7; also below).

Carrie Lane (2011) tells the story of many such workers in Dallas, Texas. She tells, for instance, the story of Alex (a pseudonym), a former information architect who “had established a reputation -locally and internationally” but now (in the autumn of 2001) worked as a waiter. Lane tells us that, even before this had happened, employers of Alex and his ilk:

“...had warned them -usually explicitly- loyalty and security were no longer part of the social contract of employment nor, for that matter, were benefits, pensions, retirement accounts, full-time work schedules or company-sponsored training. Their professional lives were instead destined to comprise a collection of part-time short-term, usually contract positions bridged by periods of unemployment, underemployment and self-employment.”

Still, these workers did not perceive themselves as victims of free market forces or corporate capitalism. They saw themselves as “companies of one” who were constantly “defining, improving and marketing themselves”. They had lost in the “new economy”, but they were not the losers. For them,

“...the losers were those outmoded men and women who failed to cast off the dependent mindset of the “organization men”, who foolishly looked to paternalistic employers to provide them with job security and financial stability.” (Lane, 2011: 9)

The apathy (if not antipathy) that these workers showed towards the “organization men” is not only the clearest indication of the fall of employment norms that were held dear just two decades before, but also a sign of the extent of fragmentation among computer workers.

But despite the appearances, the way in which workers entered the IT labor market in the new economy is reminiscent of the way in which the generation of “organization men” had entered their jobs. Back then, too, there was no shared understanding of required skills and virtually no barrier to entry. Employers were equally eager to hire anyone who could pass a test that ostensibly showed their potential for programming computers, or at least learning to program computers. Apart from the changes in employment relationship, what had changed between the two groups was the emergence and establishment of platform-based (layered) competition and a vast repository of interconnected tools and technologies that now seemed to position not only companies but also workers vis-à-vis each other. But in reality, it was changes in the regime of skill formation that had tied the fate of workers to technologies and technologies to competition between companies. Moreover, while IBM’s dominance in the



market (along with its emphasis on employment stability and security) in the previous decades had provided some measure of stability in jobs, within the span of one decade, changes in computer jobs headed towards fragmentation and even more destabilization of the labor market.

In this chapter I have shown how the unfolding structure of industry as well as decisions of companies competing for market domination led to fragmentation of IT jobs during the late 1980s and through the 1990s (see Table 8-6). In the next part, I put together the different episodes of my historical narrative to rearticulate my argument in shorter form and draw conclusions about the current conditions of IT work.

Job Title (SOC code)	1990		1993		Job Titles (SOC code)	1997		1998		Job Titles (SOC code)	1999		2000		2001	
	Employed	% of Ind	Employed	% of Ind		Employed	% of Ind	Employed	% of Ind		Employed	% of Ind	Employed	% of Ind	Employed	% of Ind
Computer Scientists & Related (25100)	212,260	26.48	240,390	27.14	Computer Scientists & Related (25100)	490,050	32.91	592,900	35.14	Computer and Mathematical Occupations 15-0000 (1)	893,860	47.14%	1,056,700	47.11%	968,950	48.75%
Systems Analysts (25102)	68,990	8.61	81,150	9.16	Systems Analysts (25102)	127,760	8.58	148,260	8.79	Computer Systems Analysts (15-1051)	136,160	7.18%	153,110	6.83%	144,330	7.26%
					Data Base Administrators (25103)	17,840	1.20	19,120	1.13	Database Administrators (15-1061)	24,730	1.30%	28,420	1.27%	26,140	1.32%
Computer Programmers (25105)	117,360	14.65	127,030	14.34	Computer Programmers (25105)	203,780	13.69	245,790	14.57	Computer Programmers (15-1021)	218,900	11.54%	220,090	9.81%	207,680	10.45%
										Computer Software Engineers, Applications (15-1031)	152,610	8.05%	211,360	9.42%	183,620	9.24%
										Computer Software Engineers, Systems Software (15-1032)	78,090	4.12%	111,830	4.99%	104,130	5.24%
Computer Programmer Aides (25108)	14,010	1.74	13,670	1.54	Computer Programmer Aides (25108)	16,700	1.12	18,960	1.12							
					Computer Support Specialists (25104)	102,170	6.86	130,700	7.75	Computer Support Specialists (15-1041)	133,680	7.05%	157,110	7.00%	146,940	7.39%
										Network and Computer Systems Administrators (15-1071)	49,030	2.59%	64,860	2.89%	58,800	2.96%
										Network Systems and Data Communications Analysts (15-1081)	28,200	1.49%	35,820	1.60%	35,200	1.77%
Other Computer Scientists & Related (25199)	11,900	1.48	18,540	2.09	Other Computer Scientists & Related (25199)	21,800	1.46	30,070	1.78							
										Computer and Information Scientists, Research (15-1011)	11,800	0.62%	9,340	0.42%	9,390	0.47%
Industry Total	779,090	100	885,790	100	Industry Total	1,451,030	100.00	1,662,750	100.00	Industry Total	1,896,070	100.00%	2,243,030	100.00%	1,987,460	100.00%

Table 8-1: Occupational employment in Computer Programming, Data Processing, and Other Computer Related Services (SIC 7370) for selected years between 1990 and 2001. SIC 7370 includes Computer Programming Services (7371), Prepackaged Software (7372), Computer Integrated Systems Design (7373), Computer Processing and Data Preparation and Processing Services (7374), Information Retrieval Services (7375), Computer Facilities Management Services (7376), Computer Rental and Leasing (7377), Computer Maintenance and Repair (7378), Computer Related Services, NEC (7379). Based on U.S. Department of Labor Occupational Employment Statistics.

**Part Three**  
**Discussion and Conclusion**

## **9 Discussion**

In part two, I have provided a narrative of changes in software work from its early days in the computation labs to the dotcom crash of 2001. In the narrative, I followed the interactions of employers and worker collectives with the state (court rulings, government actions and pieces of legislature) as well as those between worker collectives and companies (as employers) and between different companies (in the market) to provide an explanation for the continuous changes in IT work. In this chapter, I provide a summary of the account presented in the previous chapters, highlighting the key points raised in each period and how it shaped the subsequent developments. Based on this summary, I will present the argument in the subsequent section. Finally, I will discuss how this analysis can help understand the increasing fragmentation of IT work since 2001.

### **9.1 Summary of the narrative**

In the first period, between 1945 and 1954, competition was either inexistent or in its early forms. In the context of war and military need for frequent and accurate calculations, early computers were created to automate the tasks of human computers. But since using electronic computers required programming (inputting instructions) in computer-understandable form, mathematical knowledge was still required to translate formulae developed by scientists and engineers into machine-understandable form. However, male scientists and engineers saw these tasks as low-level and mechanical and were happy to delegate them to another group of workers. At the same time, because of the war, a vast majority of human computers who assisted those scientists and engineers were women. Given the prevailing attitudes of the time about the role of women, those scientists and engineers decided that women were more suitable for low-level jobs. Thus, while computers were a relatively new technology that required preparation and operation, new jobs were created around them partly because those in charge of the computer labs thought that activities involved in those jobs were rudimentary and mechanical.

Moreover, even though the variety of the computers in existence at the start of 1950s was still small, there was no agreement over the procedures of programming work, or its technics. Each lab had developed its own practices, instruction sets and programs. Nevertheless, most of programmers had substantial mathematical education and many of them were women. Therefore the variety of technological tools and work practices did not mean that they were in different jobs. This shared background helped them form a community through which they exchanged ideas and experiences.

With the foundation of EMCC as the first company aiming at commercialization of computers, the IT industry was born. By 1951, companies like EMCC, ERA and IBM had started to sell computers to businesses. Many of the new computers were bought for use in business (rather than scientific) calculations, which required new skills: companies needed individuals who would ‘analyze systems’, understand organizational processes and calculations and state the problems in ways that a programmer could then translate into computers’ language. Thus advanced mathematical skills were no longer required but there was no agreement about what other skills programmers needed to be able to program correctly and efficiently. At the same time expansion of market for computers meant that more and more programmers were going to be required.

While working at Univac, in order to reduce the pressure on her staff, Hopper developed a compiler that became the foundation of modern programming languages. After Hopper overcame initial resistance from managers and convinced her colleagues in the industry that programming a computer to program itself was both possible and desirable, programming tools and languages became gradually more widespread. While her main motivation for developing a compiler was to ‘return to be a mathematician’, employers began offering programming tools as free bundled facilities that would make computers much easier to use and reduce customers’ need for programmers. But rather than reducing the need for programmers, programming languages contributed to wider use of computers in the business context and replaced mathematics as the knowledge that was required for programming computers. At the same time, because programming and training were offered free of charge by mainframe vendors, there were very few software companies. Hence, even though the object of work was layered, there was no vertical disintegration in the industry.

This was in an environment that programmers were difficult to find and even programming trainees were highly sought after. Companies like IBM and Rand started their training programs but, by all accounts, even with enlisting universities, they were unable to train as many programmers as the industry needed. Industry and user organizations raised alarm but the federal government did not take any action. It was the SAGE project that led to establishment of SDC as the major training center for programmers. SDC also pioneered psychometric tests for selecting from among trainee candidates. At the same time, because demand for programmers was increasing every day and the question of skills had remained unresolved, employers (including SDC) gradually began to relax entry requirements. Psychometric tests become gradually the de facto criteria for entering the computer industry.

*Thus, while in the first period (until 1954) the size of labor market was small and many programmers shared educational backgrounds, in the second half software workers came from very different backgrounds. Still, the majority of software work was done by programmers, systems analysts (and other support staff) who worked in IT corporations or on mainframes at customers' site with support from IT corporations. Therefore, despite differences in their tools and practices, categories of software work were not greatly differentiated.*

Selection of IBM as the main contractor of SAGE project helped IBM use the advances made by Whirlwind scientists and engineers in its products. This and IBM's extensive investment in training its sales and programming workers and integrating them with each other enabled IBM to become the dominant company in the competition, an event that marks the start of the second period in this narrative. In 1956, IBM settled a case filed by the justice department by agreeing to, among other requirements, separate its services division as a separate organization (SBC). SBC quickly became the dominant organization in the services sector where other software and services companies had also emerged. In 1957, a group of ERA members left Univac (Sperry) to found CDC. CDC quickly became a serious rival for IBM, especially in services sector. At about the same time EDP schools began to appear. CDC later established its own chain of schools and acquired another one and became one of the major training providers in the industry.

With the entry of CDC and other companies near the end of the 1950s, competition took the form it was going to have for another two decades. Introduction of minicomputers (following the founding of DEC) and emergence of independent software and services companies both expanded the market and created a limited amount of vertical disintegration. Since vendor companies continued to provide bundled software and services, companies in the independent software and programming services sector had to compete against 'free' vendor-provided packages and services. Therefore, unlike vendor companies, which provided employment stability (if not security), independent companies had to offer stock options to attract and retain their programmers.

As continuous expansion of the market and perceived shortage of programmers attracted more and more people to computer work, EDP schools, vendors and computer science schools continued to provide training. Even the government became involved with the Vocational Education Act of 1963. This provided a variety of options for people who wanted to enter the computer labor market. They could pay for their training or have their employer (including the government) or the IT vendor pay. Towards the end of the decade,

even high school graduates had a good chance of finding jobs. Several professional associations for software workers emerged, but rivalry and disagreements between them prevented them from creating an overarching body. In 1962, DPMA began to provide its CDP certifications but, because it did not receive much support from other associations, it was not successful in the long term.

The most important event in competition in the mid-1960s was the introduction of System/360 by IBM. Following the success of Honeywell 200 in capturing market from IBM, IBM became more determined to introduce the new system as soon as possible. While several companies had been considering introduction of compatibility across their products, IBM was the first to introduce a compatible series of computers. One of the primary objectives of IBM in introducing the System/360 was to save the amount of retraining and programming that were required for providing support and training for different models. It also enabled software programs to be written once and used on several different models. Since this enabled customers to save the costs of re-programming after each computer upgrade, System/360 was hugely successful and re-established IBM's position as the dominant corporation in the market. But success of System\360 created new forms of competition for IBM. Now other vendors started producing IBM-compatible products from small parts and peripherals to full-scale computer-systems. More importantly, given the success of the series as a whole, software companies could write program packages that could be subsequently run on many computers.

*Software workers in this period were still mostly working on mainframes either in IT corporations or representing IT corporations but a small number worked in software and services companies. Its divisions and categories of work were mostly similar to the period before, but independent software companies provided the possibility of a slightly different work setting. Two alternative possibilities for software work were closed down in this period. The first possibility was organizing programming work in rigid structures and close control over it. As we saw, while cost-saving through automation became the dominant concern for IT corporations and their customers, designers at SDC chose a different approach that was more costly but provided senior programmers and managers with more control over the activities of junior programmers. It took the industry years to rediscover the control methods and tools that SDC managers developed in response to shortage of experienced programmers. If the SDC approach to programming had become widespread, programming work was likely to be a much more structured activity (with clearly-defined roles, responsibilities and ranks). It is doubtful, however, that under such "factory"-like organization of programming work, software jobs would be as creative as it is now.*

*The second possibility was maintaining a degree of research orientation in software work. Producing research and participating in conferences was typical for programmers in the earlier years, but became increasingly prerogative of a small group. As companies became more and more concerned with the costs of programming and wanted programmers to concentrate on their 'main' job, the research component of programmers' jobs disappeared. If the cost saving rationality of managers was not so dominant, or market growth and training of programmers were more commensurate and programmers could keep the 'luxury' of research, it was likely that programming work (and subsequent developments) would take a different shape.*

The third period (1969-1985) began with IBM's decision to unbundle software and services from the sale of hardware. Most of the other companies were initially reluctant to follow IBM's lead. Nevertheless, over time, more companies unbundled software and services from hardware. Like the introduction of System\360, the unbundling decision did not have immediate effects on the competition. IBM remained dominant in both hardware and software and other companies competed for the second place. The unbundling decision was, at least in part, motivated by the U.S. Department of Justice antitrust lawsuit against IBM. Even though the case was dropped 13 years later and without any apparent achievement, the lawsuit did influence decision making at IBM as IBM managers tried to avoid further escalation of stakes in the case. As we saw, in order to settle with CDC, IBM agreed to sell SBC at a low price, making CDC a dominant company in the services industry.

Potential effects of unbundling were undermined when the industry stopped growing during the recession of the beginning of the 1970s. GE and RCA left the industry. Labor market expansion momentarily halted and employment security became less certain, even though many companies tried to uphold it. When growth returned to the industry, the main beneficiary from IBM's unbundling decision was the software industry, which finally could have a chance to compete with vendor companies. Corporate software industry took shape as a handful of companies became established as the dominant independent software companies. Nonetheless, their share of the market was significantly smaller than IBM's.

Another result of unbundling decision was that vendor-provided training was no longer free. University programs continued to expand, but still a significant part of the workers were trained outside universities. Meanwhile different professional associations tried to overcome their differences by forming ICCP as a certification providing body. ICCP overtook CDP exam from DPMA but failed to make it more appealing to the workers. It was also



weakened by infighting between different associations. One of those associations, SCDP, tried to solve the problem by sidestepping other groups and submitting a bill to state-level legislatures across the country. The proposed bill, called for state to take the licensing of computer workers into its own hands. Other professional groups reacted ferociously and tried to block the bill by any means possible. The bill failed in every state that it was submitted and was gradually forgotten. As the labor market resumed growth, entry to it remained unchecked.

A few companies established a new niche by manufacturing microcomputers based on microprocessor chips. A number of computer workers who had some experience of working with mainframes and minicomputers at the industry (or universities) turned their attention to writing software for microcomputers. Many of them drew on the skills that they had gained in the relatively open labor market of the industry and the skills that had become widely available to develop the first, programming tools (like operating systems and programming languages) for other user/developers. Gradually new applications were developed for microcomputers. These developments made microcomputers (like Apple II) useful and appealing for a broader group of users (beyond hobbyists). Therefore a group of successful PC software companies emerged that created a new, distinct software industry. The newly formed PC software industry had a different business model from the corporate software companies.

IBM identified the growing opportunity and decided to enter the market using an open strategy. It purchased chips from Intel and asked Microsoft to provide the programming language and operating system. Other companies were invited to develop software for the new PC. The success of PC resulted in vertical disintegration in the industry and, following the reverse engineering of BIOS, entry of a large number of cloning companies to the market. Cloning companies rapidly curtailed IBM's market share while providing impetus for dominance of MS-DOS as *de facto* operating system for PC. Microsoft soared on the success of DOS but still it was only one of the top companies in the PC software industry.

The industry was also hit by the economic recession of the beginning of 1980s. Most of the mainframe vendors tried to avoid layoffs or reduce the effects of layoffs on employees. In contrast, most of the companies in the microcomputer industry and the software industry had recourse to layoffs without any further commitment. Some, like Apple, had explicitly refused to provide any guarantees or security. Still, IBM was dominant over the industry as a whole and therefore employment security continued to be seen as a beneficial strategy for a dominant company. At the same time, 'Apple deal' was becoming increasingly acceptable.

*In this period (1969-1985), with the exception of the economic recessions of the first years of 1970s and 1980s, labor market grew rapidly as new software companies appeared everywhere. Software work became divided between three types of producers under three different conditions. Some (but not all) mainframe vendors tried to prevent fluctuation in their headcount and protect their workers from layoffs. Corporate and PC software companies, in contrast, made no such attempt but, during the growth years, offered stock options to attract employee's loyalty. But because these two sectors of the software industry had different business models, software work in each of them was very different from the other: corporate software companies, like companies in the mainframe sector had integrated programming, support and sale services while software work in PC software companies was concentrated in software development. This latter group provided almost no customer training and minimal support. Sales and distribution of its products was often handled through third parties.*

The fourth and final period of my study began with the shakeout in the mainframe industry. Several companies left the industry and layoffs became widespread. The value of old norms of employment relationship (such as security and loyalty) was questioned. IBM started a massive restructuring program that was aimed at re-orienting the organization while preserving employment security (through retraining and reassignment). At the same time, in order to improve its financial situation, IBM made a deal with the Congress in which computer workers were explicitly excluded from the provisions of the Free Labor Standard Act. Microsoft became dominant in the PC software by successfully launching Windows 3.1 and providing Windows compatible software applications ahead of the rest of industry. It also sought to establish its position by challenging Novell's dominance in the network operating system market and, ultimately, replaced it.

IBM failed to make a comeback and IBM managers decided to abandon the no-layoff policy. Massive layoffs followed, indicating the end of the employment security in the IT industry (or even the U.S. economy). The new employment relationship that had been first practiced in software firms became the norm of industry. Remarkably, despite the recession and unemployment in the wider economy in the early 1990s and widespread layoffs in the industry, IT labor market continued to grow. Many software workers were trained through vendor-provided certification programs. Gradually this new institutional arrangement for training replaced the traditional vendor-provided arrangement (indicated by changes in both IBM and Microsoft's policy). Instead, competition in the products was extended into the realm of training (examples include competitions between Microsoft and Novell and between Microsoft and jCert Alliance).

Meanwhile, older professional associations struggled to define their relevance to the labor market as their memberships stalled and their certifications failed to attract the workers. New professional associations (like NCEPA) similarly failed to attract any significant number of workers. All these associations were absent from discussions about the rights of permatemps and the cap on H1-B visas (with the exception of IEEE about the latter issue). In the absence of support from professional organizations, some workers formed new worker organizations with the purpose of challenging corporations and lobbying the government (Washington Technology Alliance and Programmers Guild). However, their small size prevented them from having any major effect on the situation of computer workers at the industry level. When the dotcom bubble burst, industry layoffs followed and the labor market shrank significantly for the first time. Many workers found themselves struggling for survival.

*In this last period, software workers became rapidly fragmented as new areas of work emerged and attracted new workers. Existing associations were unable to control entry to the labor market or even attract the new workers. The new vendor certifications also made certifications of (older) professional associations irrelevant. Instead workers became embroiled in the competitive strategies of rival companies and the rapidly changing dynamics of market.*

## **9.2 Reconstructing the argument**

My objective in this research has been to explain why IT jobs have not been institutionalized in the way that other jobs have. As I argued in my theoretical framework, in order to be institutionalized, i.e. to have 1) established roles and responsibilities in the workplace, 2) clear boundaries and relationships with others and, 3) means to reproduce these over generations of workers through socialization and provision of skills, an occupational group must first emerge that actively engages in achieving those ends. In the absence of such a group, it would be impossible for jobs considered part of “the same occupation” to become institutionalized in the contemporary society. Moreover, I argued that emergence or absence of such a group must be understood in the historical context of its formation and the institutions that evolved around the occupation. Since IT work, like many other work areas that emerged in the mid 20<sup>th</sup> century, emerged and evolved inside organizations, I argued that staffing decisions of organizations have been very influential on its formation.

I further postulated that, while in the contemporary world organizations center their staffing decisions around the notion of skills, skills that are required for entry into an occupation become the subject of a social process in which different social groups participate and reach

a political settlement about four issues:

- 1- What are the required skills?
- 2- Who has or is best placed to earn those skills?
- 3- Who shall provide (or pay for the provision of) those skills and how?
- 4- What does earning those skills mean in terms of the rewards individuals receive?

In the earliest cases of programming work in the computer laboratories before the formation of competition (1951), the answer seemed simple enough. We saw that scientists and engineers who had designed and developed early computers believed that programming was a relatively mechanical and mundane task that nevertheless requires substantial mathematical skills (Q1). They also recognized that the best group of workers who had these skills were human computers, i.e. mathematically educated individuals who had assisted scientists and engineers in similar contexts (Q2). Moreover, while those scientists and engineers were equally able to perform programming tasks, because of the prevailing sexist attitudes of the time, such low level jobs were deemed to be 'more appropriate' for women. Implicit in this argument, is the assumption that one could not expect to earn much reward from programming jobs (Q4). At the same time, 'being trained' was part of the job and therefore training was provided for free (Q3).

For a limited time, in the subsequent years, these assumptions and answers persisted. Mathematical skills were required and only those who at least had a substantial mathematical training could hope to be selected. But gradually some of these assumptions and answers were challenged. It became clear that programming is not necessarily mechanical. If anything, it was a complex and challenging job that required special talent and certain mental abilities. But nobody knew how to identify those talents. Most importantly, with the commercialization of the computer technology the context of computer work was increasingly changing from scientific applications to business applications. Commentators agreed that mathematical skills were not so useful in the new context but there was no agreement about the new skills that were required. With the intensification of interest in computers and competition in the industry, new interests had become involved and new answers to those questions had to be found.

I had suggested in my theoretical framework that, as a society-wide issue, the answer to these questions are settled through power-induced interactions between the workers, employers and the state and the way in which these actors align with each other can shape the outcomes of settlement. I had also argued the power-induced interactions that simultaneously shape regimes of skill formation and employment relationships occur a) between different employers through competition in the product market; b) between

employees and employers in the labor market/employment relationship; c) employers, employees and the state through lobbying and appealing (by the first two) or regulation by the state. In the context of competition between companies in the American IT industry, I pointed out that workers can be assumed to be aligning with their employers in the product market because the success or failure of products depend on their skills. Therefore I highlighted the importance of the platform and object of IT work as well as the value contribution of IT workers for understanding how workers can contribute to an employer's success/dominance. Similarly, vertical integration or disintegration of the industry was important because it defined the relationship between the products of different companies and how they may influence each other.

The central argument that I have put forward in this thesis is that, in the absence of a governmental intervention aimed at stabilizing the labor market, the strategic and tactical decisions of IT corporations involved in the dynamics of IT product competition destabilized IT jobs and effectively prevented the formation of a solid occupational collectivity which is a precondition of institutionalization of jobs. As I have shown throughout my study, professional associations in software work were primarily concerned with only the first question but without ever reaching any agreement. They were also concerned about whether any one group may claim exclusive authority and therefore had deep and sometimes bitter rivalries. During the five decades of this study, only one association made attempt to create barriers to entry to the computer labor market (SCDP bill) and it failed. I could find no other example of engagement of any of these associations with questions 2, 3 or 4 at any point in the industry. At best, educational associations like ACM and IEEE tried to shape subjects of teaching and research in universities. But they did not try to become directly engaged in the in the labor market.

I have also shown that the U.S. Government did not become deeply involved in the issues of IT labor market until the last period of my study. Its involvement in the previous decades was only in the IT product market and with the objective of preventing formation of monopolies in the market (cases of IBM and Microsoft). Therefore the focus of discussion in this section will be primarily on the role of IT corporations and their interactions with workers. Towards the end of discussion I bring the state back in.

At the beginning of commercialization of computer technology, IT vendors had an interest in expanding the market and therefore made two important decisions. One was to provide software and services free of charge for customers; the other was to train the workforce for software work including programmers and systems analysts but also technically competent

salesmen and other support roles. However, their training programs were not capable of producing enough workers for the rapidly expanding market of their products. This created a sense of looming shortage crisis that never disappeared from the horizon. Throughout the 1960s, amid confusion about the nature of skills and the characteristics of a competent programmer, companies opened the gates of computer work to almost anyone who was interested in learning programming, had a high school degree and could pass a psychometric test. This diversity became a characteristic feature of the IT labor market that was never rolled back or resisted. Therefore it had a lock-in effect: those who had entered computer work as high school graduates or employers who had hired high school graduates before were unlikely to enforce barriers to entry at a later date.

At the same time, several factors prevented the need for union-like or similar organizations in IT work. First was the rapid change in the status of software work from a low-level job to one of the most highly prized jobs with seemingly infinite potential for growth. Moreover, the rapid growth of industry throughout this period meant that there was now cause for grievance or union-like solidarity. Third, several companies (including IBM, as the dominant company in the market) provided employment security and stability in a way that, not only removed any possibility of unionization, provided the industry with a norm of employment relationship. This norm was, however, more 'ideal' than practiced in the sense that the value of employment security was taken for granted, even though not always practiced. Corporate software companies and some computer companies that were unable to provide such security were seen to be in a disadvantage and tried to attract programmers and other software workers by offering stock options. In 1969, IBM's unbundling decision enabled some vertical disintegration in the industry and provided corporate software companies with some space for competition. Still, IBM remained the dominant company well into the next decade while other companies faced more challenges (especially in the beginning of 1970s) and layoffs began to appear in the computer industry.

In mid 1970s, an unlikely match -between electronic hobbyists and a group of young to middle-aged people (mostly men) who had some experience of software work in the corporate environment- transformed microcomputers from a curiosity to a usable tool. While the rest of computer industry had ignored microcomputers, these individuals began to develop them into a more complete and user-friendly device. Many of the PC software developers and entrepreneurs involved had benefited from the low entry requirements of software work in the decade(s) before, and also drew on the skills that were disseminated during that time, (particularly BASIC).

Emergence of a flourishing market for microcomputers attracted IBM. However, IBM decided to break with its tradition of secretive, internal product planning and used an open architecture for the PC. The effects of this decision (and the fact that IBM could not control the subsequent events) were twofold: 1) industry became vertically disintegrated, and 2) ultimately IBM's position as the dominant company was undermined. While, until then, IBM's dominance over the industry had protected the norms of employment relationship from changing, gradually new practices gained credibility.

These new practices were promoted by the new PC software industry and some personal computer companies (such as Apple). This was despite the fact that many corporate software companies had been using similar practices for more than a decade. But because of their business models, they were unable to take advantage of such practices to challenge IBM, let alone change the norms of employment relationship. But the new PC software companies, which had a different business model and relied on workers' skills in a different way, used these new employment relationships to their advantage. For example, programming and sales personnel at corporate software needed to have an intimate knowledge of customers' needs, often created after years of experience, while PC software companies had minimal contact with the final customers and could recruit developers straight out of college without needing to keep them. Similarly, corporate software companies used layoffs mainly as a survival strategy, while their counterparts in the PC software industry used them as tactical moves (terminating one product line and switching to another). Therefore, three distinct forms of software work existed in the early 1980s. First, those who worked with mainframe/minis in corporate environment with employment security (IBM, HP, DEC) second, those who worked with the same technologies without security (mostly in other mainframe/mini vendors and corporate software companies) and third, those who worked in PC vendors and software companies.

When Microsoft replaced IBM as the dominant company in the industry, the new employment relationships came to be associated with success and gradually became the norm. Interestingly the introduction of new employment relationships did not spark protests or collective action almost anywhere in the industry. One important reason for this could be the parallel political changes that occurred in the American society, most notably the anti-union policies of Reagan administration which *coincided* with changes in the IT industry. Therefore, it is not clear to what extent Microsoft's rise to power was influential in the legitimization of new employment relationships. Nonetheless, it is notable that, in principle, companies like Microsoft had the option of introducing protections for their employees but

they chose not to. When IBM began layoffs in 1992, it was clear that the (traditional) norms of employment security were a thing of the past. The few companies that continued to maintain their traditional employment relationships were soon forced to adapt themselves.

Moreover, change in the structure of industry meant that new occupations could emerge to take charge of the new layers of technology. Two such occupations that I briefly considered in the new work environment (network and web development) were substantially different from software development and programming and disjoint enough to form new associations, practices, etc. Networking companies, in particular, pioneered a new institutional arrangement for training that was soon used by software companies and others as well and became one of the main mechanisms of entering the IT labor market.

Moreover, in the last period of my study, we saw that IT corporations lobbied the government for company's own benefits in ways that affected software work. IBM's deal about exclusion of programmers from Fair Labor Standards Act and the pressure on Clinton administration to raise the cap were two important aspects of this. By the end of the 1990s, there is still no definite answer to the question "what skills are required for entering software jobs" but it was clear that at any point in time, the answer was determined (either explicitly or implicitly) by IT corporations rather than universities or professional associations. The second question (about who is better placed to earn those skills) has become irrelevant partly because the labor market has been historically open to anyone, and partly because it depends on the answer to the first question. The third question (about who pays for the training) has become a mixture of private and public funding for university education and private (individual's) investments in improving/certifying their skills through vendor certification and similar programs. Finally, the answer to the question what can workers in those jobs expect depends to a large extent on the exact position of the job in the industry. A stark distinction exists between the 'star developer/engineers' who receive stock options and are highly paid, and workers in the secondary labor market who enter IT jobs out of necessity or as a temporary arrangement. The majority of software workers are distributed in the spectrum between the two extremes.

Thus, while IT jobs continuously changed throughout the history of industry, it was mainly in the last decade that jobs began to differentiate. The reasons for this must be sought in the decisions that IT companies have made throughout the history of industry. By lowering the entry requirements to the IT labor market (and keeping it low) and freeing themselves from the commitments of employment relationship while enrolling workers in their competitive strategies, IT corporations have prevented the formation of an occupational collectivity.



### **9.3. Developments since 2001: ‘the present as history’**

The suggested framework and explanation can be used to shed light on aspects of the developments that have occurred since 2001. The U.S. IT labor market did not begin recovery until 2003 and did not return to the levels of employment that it had reached in 2001 until 2007. In the meantime, new companies emerged to challenge Microsoft’s dominant position, most importantly Google and Apple. But rather than challenging Microsoft at its stronghold (PCs), these two companies sought to capture the new growing markets. Thus, while Apple had an established presence in the personal computer market and Google was a search-engine start-up, both companies moved into the market for the handheld devices including tablets, smart phones.

It is notable that similar devices (like Personal Digital Assistants) had been available since early 1990s but had relatively limited functionalities (see Evans, et al, 2006: 155-181). The newer generation of devices relied heavily on broadband and wireless networks, which had been developed in the early 2000s, for their usability. Thus, a new software delivery model based on digital media distribution model was created. Customers, in this model, access applications through gateways (‘app markets’) of companies that provide the OS (or hardware) of handheld devices (see Holzer and Ondrus, 2011). While Apple has maintained its closed architecture by bundling its OS with its proprietary devices (iPhones and iPad), Google relies mainly on third-party hardware producers (even though it produces the hardware as well). Microsoft, which had a relatively successful presence in the PDA market had underestimated the potential of the new sector (Evans, et al, 2006: 215) and has been trying to catch up. Interestingly, through these developments some of the boundaries between software and hardware companies, at least at the level of major corporations, were removed. Amazon, Google, Microsoft and Apple all produce both software and hardware.

New software developers emerged to provide software for the handheld devices. Given the very wide customer-base of handheld devices and the comparatively low costs of distribution, these companies use a business model that is different from both previous software sectors. Because of the wide reach of ‘app markets’, these developers can offer their products at extremely low prices or even for free using other means (like in-app ads) as source of income (see Xia, et al., 2010). At the same time, because of the availability of various software development tools, only a small numbers of individuals are required. Some app developers work entirely alone. Therefore, there is little or no division of labor within the majority of app developers. Thus a widely scattered form of fairly homogenous software work has accompanied the rise of new ‘app markets’. Nevertheless, as expected from the findings of this study, this vertical disintegration (especially on mobile platforms) has

resulted in further fragmentation of employment relationships. Bergvall-Kåreborn and Howcroft (2013), for instance, found that majority of ‘app developers’ in their study not only “represented a diversity of employment types” but also “experienced multiple types of employment simultaneously” (formal employment and freelancing at free times, project subcontracting and freelancing, etc.).

At about the same time that handheld devices became ubiquitous, a new model of corporate software and services emerged that similarly relied on broadband networks. As more and more companies move towards “cloud computing” and using “software as a service”, large IT corporations (of different sectors) moved to capture parts of the new market. Thus, IBM, Microsoft, Amazon and Google, all are contenders in the market for cloud computing. In contrast to the highly fragmented situation of app developers, cloud workers are likely to work under conditions that are more similar to the corporate environment. Moreover, cloud computing requires ‘a new set of skills’ that none of the existing training programs have been prepared for.

A Microsoft-sponsored report in 2012 warned about the “harsh reality” of the emerging “cloud skills gap” which is (ostensibly) different from previous skill shortages:

“Unlike IT skill shortages in the past, solving this skills gap is extremely challenging, given that cloud brings a new set of skills, which haven’t been needed in the past. There is no one-size-fits-all set of criteria for jobs in cloud computing. Therefore, training and certification is essential for preparing prospective job candidates to work in cloud-related jobs.”

According to the report, 1.7 million jobs had remained unfilled in that year. Whether or not the cloud skills gap is different, the process that it is likely to unfold is imaginable. Already computer companies (including Amazon, CA Technologies –formerly, Computer Associates-, CompTIA, HP, IBM, Microsoft, Oracle, Red Hat and Salesforce.com) have started providing cloud certifications. These companies will provide certificates for yet another generation of computer workers who do not required any qualifications. At the same time, certain mathematical skills that can be used for analysis of large volumes of data have become important in the new environment. Whether or not a distinction appears between ‘mathematically-trained’ data-analysts and ‘others’ can be an interesting development to follow.

Meantime, within the political domain, the lobbies for more immigrant workers have repeated themselves. In February 2007, in an opinion piece in Washington Post titled “How to keep America Competitive” wrote that, in order to remain innovative, the United States

must “make it easier for foreign-born scientists and engineers to work for U.S. companies.” In the March of following year, as quoted at the beginning of this text, he appeared in the Congress to personally deliver the same message. Shortly afterwards, recession hit the American economy. In early 2009, Microsoft laid off 5,000 employees, its first massive layoffs in history, citing decline in the sales of Windows OS. It took Microsoft 4 years to return to the same level of employment in the U.S. Microsoft’s layoffs were, of course, only one of the waves of layoffs that passed through the industry. Its significance however was in indicating the effects of changes in the competition (Even Google laid off 100 employees in the same year.)

	Jul. 2006	Jun. 2007	Sep. 2008	Dec. 2008	Apr. 2009	Dec. 2009	Sep. 2010	Sep. 2011	Sep. 2012	Dec. 2012
Washington	33	35	40	41	41	39	40	40	41	41
U.S.	44	47	56	57	56	53	53	54	56	57
World	71	78	94	95	95	88	88	90	94	97

Table 9-1: Microsoft employment at Washington State, the U.S. and worldwide (in thousands)

More recently, as indicated above, cloud skills have become the focus of industry. Writing in the Wall Street journal, Gary Beach, a former publisher of CIO Magazine and authority on “U.S. technology skills gap”, reported:

“The country’s technology skills gap is widening, with too few people qualified for a growing number of information technology jobs....In Seattle, that tech job market has gone to the dogs. Literally. To compete for tech talent in a region dominated by Microsoft Corp. and Amazon.com Inc., Rover.com, a dog sitting service, offers employees a \$1,000 coupon, redeemable for a dog of their choice, for Web developer referrals that are hired by the firm and stay 90 days. Scott Porad, chief technology officer at the firm says “Puppy Promo” has been a success on two fronts: “it attracts talented tech workers and helps us retain them.””

He goes on to quote many CEO and CIOs who describe the shortage of cloud skills as, among other things, ‘the single biggest barrier to the future’ of industry. What was completely absent from the discussion was any possibility that companies have any plans to invest in those jobs and train individuals. Some interviewees expressed hope that recent legislations about the supply workers in ‘STEM fields’ would help resolve the problem

Following years of lobbying by Bill Gates and other businessmen, debates about the training and education of American workers have focused around the STEM (Science, Technology, Engineering and Mathematic) fields. The shared premise of all those who participate in the discussions is that future growth and prosperity of American economy depends on existence

of workers skilled in STEM fields who can bring innovation. Moreover, they believe that a looming shortage of skilled workers is endangering America's future. Two pieces of legislature in 2013 and 2014 were directly or indirectly related to the same issue (The STEM Jobs Act, and The Workforce Innovation and Opportunity Act).

Even though the label is new, such predictions, as we know, go at least as far back as 1950s. Indeed, there are detractors that argue that the shortage is a myth. While the results of these interactions remain to be seen, it is obvious that, even after the financial crisis, the balance of power in employment relationships and dynamics of skill formation has not changed. While certain advocacy groups have become more active in voicing the concerns of workers, the IT labor market remains as open and fragmented as it has ever been. It is safe to assume that, as long as the dynamics of industry and competition remain the same and norms of employment relationship don't change, it is unlikely that IT jobs become institutionalized in the near future. But to understand the significance of this we need a bigger picture. This is the task that I will turn to in the final chapter.

## **10 Conclusion**

### **10.1. Summary of argument**

In this research, I have tried to provide an explanation for the question “Why IT jobs have not been institutionalized in the way that other occupations have”. In part one, I posited that, in the twentieth century, jobs could only be institutionalized if an occupational collectivity was formed to reproduce their practices, role and responsibilities and relations with others. I argued that a historical study that focuses on the political economy of skills and employment relationships and changes in the competition can provide an explanation for changes in IT work. I identified the main actors that were to play a part in the narrative (corporations, worker collectives and the government) and argued that the dynamics of skill formation and employment relationships are shaped by interactions between these actors.

In my historical study, I followed the interactions between workers, employers and the government across different levels and in different spheres. I showed that after the early years of 1950s, in the second period, IT corporations’ push for expanding their market and their customers’ need for workers lowered the entry requirements for IT work, an effect that was locked-in in the subsequent periods. In the third period, IBM’s decision to use an open architecture for PC led to two changes in the competition. Firstly, the structure of industry and the form of competition changed and a layered and vertically disintegrated structure emerged. Secondly, IBM lost its position to Microsoft, even though Microsoft never gained the degree of dominance that IBM had. With the rise of these new companies, the employment relationship that was promoted by IBM fell and a new employment relationship became dominant in the industry. While it is often suggested that the new employment relationship reflects uncertainties inherent in the competition, I have suggested that the link between the two is more coincidental or opportunistic than real. Even before the changes in the competition, software companies did not provide employment security or (employee loyalty) but this was in most cases a survival strategy rather than a source of advantage. Moreover, if the government had intervened in the labor market in a way that protected the interests of workers (like it had done during the Second World War), a more stable labor market could have been created. At the very least, Reagan’s presidency contributed to undermining of power of worker collectivities and facilitated transition to the new employment relationship for employers. While software workers became more politically active in the last decade, I have shown that their fragmented structure and their disconnection from each other inhibited their influence over both the government and the employers.

Moreover, I have emphasized throughout that the success of different companies at different stages depended on access to skilled workers. From Rand's acquisition of EMCC and ERA, to IBM and CDC's expansive training programs, to Microsoft, Novell and Sun's certification programs, all were aimed at creating or gaining access to skilled individuals who would develop, promote and/or support vendors' products. Even in the case of emergence of microcomputers, I suggested that the low entry requirements of the previous periods and widespread dissemination of skills contributed to the possibility of transformation of microcomputers into user-friendly computers.

I have also shown that various forms of worker collectivities including professional associations, unions, guilds and the like have been largely ineffective in having any impact on the labor market. Especially, the failure of early associations to overcome their differences seems to have had long lasting effects on the formation of new collectives: new associations may be created every month, but they are unlikely to take the shape of a collective mobilization effort. While IT companies cannot be held responsible for the failure of IT associations to overcome their differences, it is almost certain that if they had reached an agreement, IT corporations would have resisted any effort to, for instance, control the labor market. This was most clear in the final years of the last period, when IT companies lobbied for, and succeeded in, raising the cap on immigrant workers.

## **10.2. Placing IT work in its social and historical context**

C.W. Mills (1959: 143) wrote long ago that "social science deals with problems of biography, of history, and of their intersections within social structures". I tried at various points in this study to shed light on how the lives of individuals were transformed as a result of changes in the fortune of IT jobs. As I mentioned in the introduction, IT jobs were the forerunners of the changes that have happened in the American economy in recent decades. Therefore in order to understand what is truly novel and what is not, and what is significant and what is not, we need to place the discussion in its broader social and historical context. In the words of Mills, "our very statement of what-is-to-be-explained" has to be placed in "the fuller range that can be provided only by knowledge of the historical varieties of human society" (p. 146). Therefore, in this section I provide some brief comparisons with other types of work or similar work in other contexts.

One aspect of comparison that can be considered is the shortage of workers. Nursing seem to provide an appropriate comparison, because it is a category of skilled jobs that have been traditionally populated by female workers who had an inferior status than their male

superiors (doctors). Moreover, there has been continuous shortage of nurses in the U.S. history. They also have a much longer history of professional associations. But it was during the Second World War that they found their contemporary organization:

“Despite the rise of paperwork and high-technology treatment, physical caring for the human body remains the center of nursing. For nurses, the great twentieth-century changes concerned not technology but conditions of employment. First, around World War II began a major shift of registered nurses from job-by-job contracts (so-called “private duty”, whether in home or hospitals); to direct employment in hospitals; prior to that time, nursing students had done the major part of general in-hospital nursing. Second, nursing differentiated into multiple levels and specialties, with registered nurses serving as bosses and intermediaries among patients, physicians, subordinate workers and hospital management. Third, nursing work itself incorporated more and more machine tending and record keeping. The generality of nursing skills, thus the replaceability of one nurse by another declined dramatically....In the great health-care expansion of the 1960s and thereafter nurses and other hospital workers began to organize and strike after decades of political passivity. [Later a division opened up between private and public nurses.]” (Tilly and Tilly, 1994: 62-3)

While nursing has a much longer tradition of professional associations, it seems that the main changes in its work were roughly contemporaneous to early changes in software work. The differences can hardly be more striking. It is also notable that partly as a result of their collective organization, nurses were successful in 1) making the National Council Licensure Examination (NCLEX-RN) a necessary requirement for foreigners who wanted to practice nursing in the U.S., 2) separating entry requirements of nonimmigrants who wanted to practice nursing (H-1A) from other categories of work (H-1B) and impose further restrictions, 3) stopping the government from extension of H-1A in 1997, making the visa category expired. The legislation that replaced it was much more restrictive (see Money and Falstrom, 2006).

The issue of unionism can be another point of contrast. We have seen that American IT workers avoided unionism, and we know that unionism even among the blue-collar American workers have never been very strong. The historical peak of union membership among American workers, which was reached in 1950s, was 35% (Sweet and Meiksins, 2008). However, this probably had more to do with the broader context of work than the skilled-nature of task. To take context to the other extreme, Friedman (1992) reports that, as late as 1988, near 70% of IT people in Denmark were union members. Moreover, a

substantial proportion of them were members of a specific union (called PROSA) that was dedicated to IT work and IT workers. While examining whether or not this feature had an effect on the trajectory of jobs in Denmark is beyond the limit of this research, it suffices here to mention that according to Friedman (1992), a large proportion of computer workers in Denmark at the time worked in service bureaus but there were few large software houses. For Friedman, this means that although the overall software industry is more dependent on imported software (especially from IBM), there are less turbulence and fewer interdependencies. Still, he points out that PROSA (and other unions) have not been able to define entry requirements. Nevertheless, their main role is training and education as well as employment protection.

These summary comparisons can help distinguish the particularities of software work in the American IT industry. The experience of nursing workers suggests that, against all the odds, if collective organization had emerged in IT work it was likely that they could succeed, at least in establishing internal specialties and hierarchies. This could also be attractive for employers because there would have been a more clear line of responsibility. On the other hand, the Danish context shows that even concentrated collective efforts were unlikely to succeed in pinpointing or fixating entry requirements to IT jobs. While evidence on this issue is not very substantial, it points out at possibility of comparative studies. In particular a comparative study of U.S. and Scandinavian countries and Japan can help refine the explanation and model.

### **10.3. Learning lessons**

Throughout this study, I have purposefully refrained from suggesting the ways in which government could have intervened to change the course of developments in IT work. This is partly because the theoretical approach of this study is concerned primarily with the ‘style’ (more or less constant historical role) of government in the administration of economy rather than any single action.<sup>1</sup> The state has a less active role (as either arbitrator, legislator or mediator) in “liberal market economies” like U.S. than “coordinated market economies” like Japan and Germany (Thelen, 2004) and we saw that this was more or less true in this study, where most important government interventions were only aimed at breaking up monopolies rather than regulating labor market or mediating between workers and employers. It is possible to suggest that if Reagan administration had adopted a less hostile approach to unions or if Clinton administration had resisted the pressure from IT corporations the fate of

---

<sup>1</sup> This is true even in the cases where an action was taken. For instance, historians still disagree whether or not government pressures on IBM had any *decisive* effect on its decision to create PC with an open architecture and by relying on outside vendors. Political lineage of commentators is often a



IT occupations could have been (at least slightly) different. However, such arguments are only one step away from wishful thinking.

However, it is possible to suggest policy options that -if taken up- could (and still can) have far-reaching effects. These policy options are not only relevant for the American context, but also for developing countries which often look up to the U.S. and want to create an innovative IT industry as an engine of growth based on the American model. It may seem that the key to success of the American IT industry has been the existence of an unregulated labor market and widely available skills. But it would be a mistake to think that developing countries can imitate this success by simply deregulating their labor markets and establishing vocational and higher education schools and programs. As this study shows, there has been a fundamental shift in the way skills are provided and employers treat workers. In the formative years of American IT industry, it was IT corporations (led by IBM) who heavily invested in the creation of a skilled workforce, long before the federal state became involved in active funding of research and training in universities and similar institutes. Moreover, there was a fierce competition between companies over the limited number of skilled workers that were available and therefore companies provided relatively generous benefits and protection measures. This made most skilled computer jobs excellent opportunities with seemingly infinite prospect of (material and technical) growth.

However, as we have seen, dynamics of competition along with cultural and political changes in the wider environment gradually but decisively transformed the political economy of skills. In the post-IBM world of IT industry, (prospective and employed) workers not only lost their employment (or employability) security but also had to pay for their skills which were increasingly commodified and sold as 'certificates' by IT corporations. This, within its historical context, contributed to creation of a fragmented labor market marked by inequality and instability.

Governments wishing to avoid the negative consequences mentioned above can adopt several options. One obvious option is trying to regulate the labor market, especially in terms of employment relationship. But if (as in the case of U.S.) this is not realistically possible, they can try to restore the responsibility of investing in skills with employers. There are several mechanisms for implementing such a policy, including: 1) obliging employers to directly invest in the training of their current and prospective employees, 2) encouraging employers to invest in training programs in universities and vocational schools,

---

very good predictor of their judgment.

3) regulating certificates by ensuring that IT companies do not simply use them as a source revenue and that IT workers receive appropriate levels of skills. It must be noted that the developments mentioned above occurred at the same time that corporations expanded their reach far beyond the borders of U.S. and were ready to employ skilled workers from anywhere in the world. Therefore, governments of developing countries must ensure that skills that they invest are useful (can be productive) within their own domestic market, independent of foreign and multinational employers. Failing to do so would mean that their investments in skills would simply provide subsidized skilled workers for multinational corporations, which can result in brain drain and stagnated industry growth.

#### **10.4. Thinking about future**

If, as I mentioned before, our understanding of past is fraught with uncertainties, using it to make generalizations and predict the future is surely very risky, if not foolish. Yet, part of the relevance of social (especially historical) research lies in its ability to relate to the pressing issues of today and competing visions of future. Therefore, one has no choice but to ask: what does this study tell us about IT work in other contexts? How does this study enhance our understanding of other occupations, especially those in which IT plays a crucial part (e.g. engineers)? What are the likely scenarios for the future of IT work? What about the future of work generally, especially if IT is likely to infiltrate (even more) all aspects of working life? These are the questions that I now try to address, with the admittedly limited analytical resources that this research provides.

This study has been largely confined to the study of software work in the IT industry to the exclusion of all other forms of IT work, especially those in user organizations. As I have mentioned before, these organizations more or less shared (participated and competed over) the same labor market that IT corporations shaped. Therefore, IT work in these organizations has surely been influenced by the dynamics explained in this research. Still, some of these organizations (especially those in services like financial industry or management consultancy) have had their own ways of structuring and rewarding IT workers. In many cases, IT skills of those workers were combined with other skills (financial in the former and managerial in the latter) One can conjecture that workers in these industries had more stability and career options than those in the IT industry. The stability of their jobs depends probably more on stability in the financial markets (as evinced by what happened in the aftermath of financial crisis). Given the rewards and prestige of their work, it is unlikely that this group of workers form a collective organization.

What does this study tell us about other occupations like engineering? The challenge for these workers -who usually have a shared educational background- lies in the way in which IT is used to commodify their skills and the extent to which their skills depend on technology. If they can protect the core of their skills or achieve exclusive rights to legitimate use of the software in the workplace, they enjoy a more or less secure position in the labor market (see Leonardi, 2012). To oversimplify the case using an example, the acid test of whether civil engineers will continue to dominate structural engineering of buildings and constructions is whether knowing a structural engineering software is enough to enter the labor market or whether, for technical or social reasons, one has to pass through civil engineering schools and receive relevant qualifications (and in some cases licenses) to be able to work. Technological innovations alone cannot be decisive in deciding between these alternatives, which are inherently social choices.

Other occupations like office workers have less chance of maintaining their task area precisely because their work requires less independently identifiable skills and, they have not been able to form new forms of collective organization that can protect their interests (after the demise of unions). Those of them that are considered as IT worker (such as call center workers) usually have a limited skill set which can be easily replaced and also provides limited opportunities for growth. This is why the contemporary debate about the implications of automation (and robots) for future of work, most directly concern this group along with manual workers.

Within this context, it is not easy to make a prediction about the future of IT work. As I have mentioned several times, it is not entirely clear which jobs belong to this category and which jobs don't. There is a distinct possibility that the same cycles of appearance of demand for 'new' skills, expansion of IT labor market and eradication of certain job categories repeat in the future. Still, one cannot help but wonder whether the category of "IT work" (at least as far as it is concerned with software) in the future can retain even the very blurred meaning that it has today. One possible scenario is that IT skills become so widespread and IT artifacts so ubiquitous that everybody 'does' its own software job (in the sense of piece of work) but very few will consider software as their 'job'. Comparison with cooking today is useful: There will be 'chefs' and other-related specialists but they will be required for very specific purposes compared to today. Most people either do it themselves or buy ready-made pieces.<sup>2</sup> Still it is unclear whether the relatively small group of workers that will be 'IT chefs' will share any common skill set. They could equally be highly idiosyncratic

---

<sup>2</sup> Signs of developments in this direction can be seen in the fact that software performance today is

individuals with very specialized skills who are either employed in a very well-paying and long term job, or hired temporarily for very specific tasks. Therefore, it is unclear will be able to form a collective that reproduces their skills and represents their interests.

This leads to the final question: are we going to see the same pattern of fragmentation in other areas of work. Is working as an individual or a member of highly decentralized and disconnected associations the trend for future of work in society. We have seen that, in recent years, long-established professional associations (in law and medicine, for example) are under strain to justify and defend their special status in a workplace that is increasingly populated by ‘knowledge’ workers. They have not been immune from trends towards individualization of work either. Is this an accelerating trend that is going to exist for some time or will there soon be a point that individuals find new foundations for forming occupational collectives and begin to reverse this trend? Clearly, I cannot provide a definite answer. What I can say is that whatever the case, both outcomes will only be achieved through a heavily contested process in which corporations (especially IT corporations of the type I have studied in this study) will have significant interest and influence.<sup>3</sup> Perhaps the way to shape the future of occupations is through reconsidering the way corporations operate and affect other parts of society.

### **10.5. Contributions**

This research has contributed to the research on IT work by providing an explanation for its continual fragmentation. In doing so it has also shed light on some darker corners of history of IT work, bringing into view the lives of some of the ordinary workers. More importantly it has provided a theoretical framework for understanding changes in skilled work that relies on organizations. Defined broadly, this category can include changes in similar areas of white-collar work like technicians and engineers.

My research also points at some of the issues that directly concern IS and other university departments that are concerned with provision of appropriate skills to their students. My research suggests that the distance between university departments and dynamics of competition is to such an extent that it is unlikely that university programs can be influential on the developments at macro level. At best, they can hope to retain their relevance by

---

usually a matter of ‘taste’ rather than efficacy.

<sup>3</sup> Of all the elements of Daniel Bell’s vision for the post-industrial society, I find this statement the most salient for the present discussion: “The private corporation in the capitalist society [...] is bound to remain the major *organizational mode of society* [in the post-industrial era] (Bell, 1973/1999: 42, emphasis mine, see also p. 269-298).

providing a more egalitarian route to IT skills that enables students to be more adaptive learners. At the same time, there has been an increasing trend in some universities to provide vendor-certifications (especially SAP) as university degrees. If this model succeeds in establishing its presence in universities, the remaining leeway for independent thinking and teaching in IT schools will be eroded. In terms of national level policies, governments hoping to overcome skill shortages must balance the employment relationships and redesign the training schemes that exist in IT work. Ultimately, faster and unequal growth is less sustainable than a slower but more balanced approach.

#### **10.6. Limitations and further research**

This study has relied to a large extent on the material that has been available from various sources on the web. This was justified in terms of the scope of the research and its empirical focus. Nevertheless, more focused and in-depth studies of individual cases may bring to light a more complicated story that is suggested here. One category of IT work that I have not considered in any depth is database, design, development and administration. Although this is justifiable because my focus had been on software work in IT corporations (rather than IS work in user corporations), a study of role IT corporations in emergence of those jobs in IS organizations can potentially provide interesting insights and perhaps corrections to the argument put forward here. Most importantly, focusing on regular employees in large corporations, I have ignored (underemphasized) the role of freelance and temporary workers, communities of practice and hiring agencies. More focused studies can clarify the role of these actors and provide a more complete picture.

**Appendix**  
**Event Map of the Historical Narrative**

1945-1950		Timeframe		1951-1954				
Harvard Mark I	ENIAC	Scientific/Military Sites						
Electromechanical	Digital	Technological Platform	Common	Mainframes				
			Innovations		Hopper's compiler			
		Software Delivery Practices		Bundled software and support services				
Hived off by scientists	Hived off by scientists	Software Work		For corporations in/by mainframe vendor companies				
Mathematics		Training	Industry	University graduates, mostly in mathematics or physical sciences.				
			Company	Training for salesmen and programmers				
	EMCC founded to commercialize computers	Companies/ Organizational Decisions		IBM enters the industry	Rand acquires EMCC and ERA		SAGE project contract is awarded to IBM	
		Hiring Decisions/ Entry Criteria	Company				Psychometric tests	
			Industry	Combination of degree and aptitude tests.				
		Employment Relationships		Shaped by the requirements of war and post-war agreements between employers and workers.				
		Competition	Rand dominates the industry					
			No vertical disintegration. Changing context of programming towards business applications.					
		Labor Market	Small community of programmers.	1954 Wayne conference raises the issue of computer worker shortage.				
		Worker Collectives	ACM, founded in 1947, was primarily oriented towards academic and scientific concerns.					
War, Dominant sexist attitudes among scientists and engineers.	Socio-political Environment		Cold war					
Investment and support for computer projects	State Actions/Regulations					SAGE Project		

Event map of the first episode (1945-1954)

Timeframe		1955-1969						
Scientific/Military Sites		SAGE						
Technological Platform	Common	Mainframes + Minicomputers  Operating Systems and Programming Languages (and Proprietary Data Communications)						
	Innovations				Minicomputers			Compatibility norm
Software Delivery Practices		Bundled software and support services + Programming services, packaged programs and data processing services						
Software Work		Mostly for corporates in/by vendor or service companies + limited amount in corporate software companies						
Training	Industry	Vendor-provided training, EDP schools, Computer Science departments.						
	Company	Programmers' University			(Later) Control Data Inst. and C-E-I-R Automation Inst.			
Companies/ Organizational Decisions		RAND (later SDC) joins SAGE	Software/services companies begin to emerge	IBM establishes SBC	ERA members leave Univac to found CDC	DEC founded		IBM reacts to by speeding up S/360
Hiring Decisions/ Entry Criteria	Company	Trainees selected by psychometric tests						
	Industry	Science degree requirements dropped. Psychometric tests become the standard test for entering programming jobs. Towards the end, High school degree becomes sufficient.						
Employment Relationships		IBM and a few others offered employment security. Other vendors did not but there waer no layoffs because the industry was expandin . Software companies which depended on availability of projects offered no security but stock options.						
Competition	Company	IBM becomes dominant.						Honeywell 200 threatens IBM market
	Industry	Emergence of software and services companies leads to a limited vertical disintegration, while minicomputers expand the market.						
Labor Market		Labor market expands rapidly.						
Worker Collectives	Company/ Association						DPMA's first certification exam	
	Industry	Several professionl associations emerge or re-orient themselves towards computer work but disagreements and competition ensues among them.						
Socio-political Environment		Civil rights movement						
State Actions/Regulations				1956 IBM consent degree			Vocational Education Act	

Event map of the second episode 1955-1969



Timeframe		1969-1985									
Scientific/Military Sites											
Technological Platform	Common	Mainframes and Minis   Operating Systems and Programming Languages (+ Limited Networks)									
	Innovations					Microcomputers	IBM PC				
Software Delivery Practices		(Unbundled) Software, Programming services and data processing services									
Software Work		A large part for corporates in/by vendor or service companies + corporate software companies + PC software companies (for consumers and companies of all size)									
Training institutions		Vendor-provided training for customers no longer free. Computer science and other academic programs expand but still a significant part of workers are trained outside universities.									
Companies/ Organizational Decisions				CDC acquires SBC from IBM		Micro vendors/ software companies emerge.					
Hiring Decisions/ Entry Criteria		Similar practices continued.								Psychometric tests abandoned.	
Employment Relationships	Company					'Apple deal'					Widespread layoffs
	Industry	Employment security becomes uncertain but some companies try to uphold it. 'Apple deal' increasingly becomes acceptable.									
Competition	Sectors/ Companies	IBM unbundles software and services.		CDC nears dominating the service industry.			IBM adopts open strategy for PC.		IBM clones begin to appear.		PC Software industry takes shape.
	Industry	Unbundling allows vertical disintegration but IBM remains dominant in both hardware and software markets. Other vendors compete for second place. Corporate software industry takes shape. Economic recession hits the industry but has limited effects.				Microcomputer manufacturers create a niche.	Vertical disintegration in the industry. IBM briefly dominates the PC market but it is soon challenged by the cloning companies.		Microsoft gains power by providing DOS. Software industry divided between corporate and PC market. Economic recession hits the industry.		
Labor Market			Labor market growth halts.	Labor market resumes growth.							
Worker Collectives	Company/ Association				SCDP submits licensure bill.						
	Industry	ICCP formed by a coalition of various associations to provide certification, but infighting between different associations makes the newly formed body ineffective.									
Socio-political Environment			Economic Recession								Economic Recession
State Actions/Regulations		IBM antitrust lawsuit filed.		IBM settles case with CDC				IBM antitrust case dropped.			

Event map of the third episode 1969-1985

Timeframe		1986-2001									
Scientific/Military Sites											
Technological Platform	Common	Mainframes, Minis, Microcomputers (PCs) + Workstations (Later networked together)   Operating Systems, Programming Languages + CASE Tools   Networks and the Internet									
	Innovations			Wintel standard							
Software Delivery Practices		Corporate software and services + PC software packages									
Software Work		Divided between work for corporates in vendor, services or corporate software providers   PC software companies   networking services   'new media' companies									
Training	Industry	Similar to previous period				Certification programs become an alternative to traditional vendor-training and university degrees.					
	Company			Microsoft University relies on exant practices	Novell certification program	IBM changes training policy/ organization.	jCert Alliance formed to promote pure Java (later).				
Companies/ Organizational Decisions		Burroughs, Sperry merge. Honeywell leaves. AT&T acquires NCR.				IBM restructuring					
Hiring Decisions/ Entry Criteria	Company					Layoffs at IBM					
	Industry	Certificate-based hiring becomes acceptable. University qualifications still not required. Increasing reliance on temporary workers.									
Employment Relationships	Company	IBM retraining programs				IBM abandons no-layoff policy		Microsoft temps' controversy			Industry-wide layoffs
	Industry	Widespread layoffs. Traditional norms of security and loyalty challenged.				New employment relationship becomes established.					
Competition	Sectors/ companies	Mainframe industry shakeout and restructuring		Microsoft dominates PC software	Novell dominates the Network OS market later		Sun releases Java programming language				
	Industry	IBM loses dominance to Microsoft but because of vertical disintegration dominant companies only dominate specific layers. Competition from overseas companies becomes a major influence on the industry.									Dotcom bubble bursts.
Labor Market		Despite recession, layoffs and restructuring efforts, labor market continues to grow.									Labor market shrinks.
Worker Collectives	Company/ Association				NCEs form association			Washington Tech Alliance	Programmers' Guild		
	Industry	While older professional associations continued to be active new associations were added without any collective unity.									
Socio-political Environment											
State Actions/Regulations			FLSA excludes computer workers				Internet Privatization		Cap on immigrant workers raised.		

Event map of the fourth episode 1985-2001

## References

- Abbate, J. (2000). *Inventing the Internet*. MIT Press.
- Abbate, J. (2012). *Recoding Gender: Women's Changing Participation in Computing*. The MIT Press.
- Abbott, A. (1984). Event sequence and event duration: colligation and measurement. *Historical Methods*, 1–13.
- Abbott, A. (1988). *The system of professions: An essay on the division of expert labor*. University of Chicago Press.
- Abbott, A. (1989). “The New Occupational Structure: What are the Questions?”. *Work and Occupations*, 16(3), pp.273–291.
- Abbott, A. (1991a). The future of professions: occupation and expertise in the age of organization. *Research in the Sociology of Organizations*, 8(1), p.17–42.
- Abbott, A. (1991b). The order of professionalization: an empirical analysis. *Work and Occupations*, 18 (4) 335-384.
- Abbott, A. (1992a). An old institutionalist reads the new institutionalism. *Contemporary Sociology*, 21(6), 754–756.
- Abbott, A. (1992b). From causes to events. *Sociological Methods & Research*, 20(4), 428–455.
- Abbott, A. (1993). The sociology of work and occupations. *American Review of Spciology*, 19, 187–209.
- Abbott, A. (1997). Of Time and Space: The Contemporary Relevance of the Chicago School. *Social Forces*, 75, 1149–1182.
- Abbott, A., (2005). Sociology of work and occupations. In Neil Smelser and R. Swedberg (eds.) *The Handbook of Economic Sociology, 2nd Edition*, Princeton University Press.
- Abbott, A. (2008), ‘Professions, Sociology of’, in N.J. Smelser and P.B. Baltes (eds.) *International Encyclopedia of the Social and Behavioral Sciences*, pp. 12166-12169, Elsevier.
- Adams, T. L. (2007). Interprofessional relations and the emergence of a new profession: Software engineering in the United States, United Kingdom, and Canada. *The Sociological Quarterly*, 48, 507–532.
- Akera, A. (2002). IBM's early adaptation to cold war markets: Cuthbert Hurd and his applied science field men. *Business History Review*, 76, 767–803.
- Alagaraja, M. and Li, J. (2014). Utilizing institutional perspectives to investigate the emergence, rise, and (relative) decline of corporate universities. *Human Resource Development International*, 18 (1), 4-23.
- Aldrich, H. E. and Fiol, C. M. (1994). Fools rush in? The institutional context of

- industry creation. *Academy of management review*, 19, 645–670.
- Allan, R. A. (2007) *A History of the Personal Computer: the People and the Technology*. Allan Publishing
- Anderson, M. (1990). (Only) White men have class: Reflections on early 19<sup>th</sup>-century Occupational Classification Systems. *Work and Occupations*, 21 (1) p. 5-22.
- Anderson, R.E., and Mortimer, J.T. (1977). Sociology of computer work: Introduction, *Sociology of Work and Occupations*, 6(2):131-8.
- Archer, M., Bhaskar, R., Collier, A., Lawson, T., & Norrie, A. (1998). *Critical Realism: Essential Readings* (pp. 1–781). London: Routledge.
- Aspray, W. (2004). The Supply of Information Technology Workers, Higher Education and Computing Research: A History of Policy and Practice in the United States. In R. Coopey (Ed.), *Information Technology Policy: an International History*, pp. 54–96. Oxford University Press.
- Aspray, W., Mayadas, F., and Vardi, M. Y. (2006). Globalization and Offshoring of Software: A Report of the ACM Job Migration Task Force. *Report of the ACM Job Migration Task Force, Association for Computing Machinery*.
- Astrahan, M. M., & Jacobs, J. F. (1983). History of the Design of the SAGE Computer-The AN/FSQ-7. *Annals of the History of Computing*, 5, 340–349.
- Attewell, P. (1992a) Skill and Occupational Changes in U.S. Manufacturing. In P. S. Adler (ed.) *Technology and the Future of Work*. Oxford University Press.
- Attewell, P. (1992b) “What is Skill?” *Work and Occupations*, 17 (4) p. 422-448.
- Backus, J. (1980). Programming in America in the 1950s- Some Personal Impressions. In N. C. Metropolis, J. Howlett, & G. Rota (Eds.), *A History of Computing in the Twentieth Century: A Collection of Essays* (pp. 125–135). Academic Press.
- Baldwin, C. Y., & Clark, K. B. (2000). *Design Rules, Vol. 1: The Power of Modularity. The power of modularity* (Vol.).
- Baldwin, C.Y. and Woodard, (2009) The architecture of platforms: a unified view, in A. Gawer (ed.) *Platforms, Markets and Innovation*, Edward Elgar.
- Banville, C., & Landry, M. (1989). “Can the field of MIS be disciplined?” *Communications of the ACM*.
- Barley, S. R. (1986). Technology as an Occasion for Structuring : Evidence from Observations of CT Scanners and the Social Order of Radiology Departments. *Administrative Science Quarterly*, 31(1), 78–108.
- Barley, S. R. (1988). Technology, power, and the social organization of work: Towards a pragmatic theory of skilling and deskilling. *Research in the Sociology of Organizations*, 6, 33-80.
- Barley, S. (2008), ‘Coalface institutionalism’, in Greenwood, R., Oliver, C., Sahlin, K. and R. Suddaby (eds.), *The SAGE Handbook of Organizational Institutionalism*, pp. 491-518, London: Sage.

- Barley, S. and Kunda, G. (2004). *Gurus, Hired Guns and Warm Bodies: Itinerant Experts in a Knowledge Economy*. NJ: Princeton University Press.
- Barr, A. (2000) *Proudly Serving My Corporate Masters: What I learned in Ten Years as a Microsoft Programmer*. Writers Club Press.
- Bartlett, K.R (2004) *The Signaling Power of Occupational Certification in the Automobile Service & Information Technology Industries*, National Research Center for Career and Technical Education, University of Minnesota.
- Bashe, C. J., Johnson, L. R., Palmer, J. H., & Pugh, E. W. (1986). IBM's Early Computers (pp. 1–735). London: The MIT Press.
- Batt, R., Christopherson, S., Rightor, N. and van Jaarsveld, D. (2001). *Net Working: Work Patterns and Workforce Policies for the New Media Industry*, Economic Policy Institute, Washington D.C.
- Baum, C. (1981) *The System Builders: The Story of SDC*. System Development Corporation.
- Bauman, Z. (2013). *Wasted lives: Modernity and its outcasts*. John Wiley & Sons.
- Bell, D. (1973/ 1999). *The Coming Of Post-industrial Society*. Special Anniversary Edition, NY: Basic Books.
- Bemer, R. W. (1969). A Politico-Social History of Algol. *Annual Review in Automatic Programming*, 5, 151–237.
- Benington, H. D. (1983). Production of large computer programs. *Annals of the History of Computing*, 5(4), 350–361.
- Benner, C. (2002). *Work in the New Economy*. John Wiley & Sons.
- Benner, C. (2003). “Computers in the Wild”: Guilds and Next-Generation Unionism in the Information Revolution. *International Review of Social History*, 48(S11), 181–204.
- Bennett, A., & George, A. L. (1997). *Process tracing in case study research*. MacArthur Program on Case Study Methods, 17-19 Oct. 1997.
- Bennett, A. (2010). Process Tracing and Causal Inference, in Henry Brady and David Collier, eds, *Rethinking Social Inquiry: Diverse tools, shared standards*, 2<sup>nd</sup> edition. Rowman and Littlefield.
- Bergvall-Kareborn, B., & Howcroft, D. (2013). "The future's bright, the future's mobile": a study of Apple and Google mobile application developers. *Work, Employment & Society*, 27(6), 964–981.
- Beyer, K. (2009). *Grace Hopper and the Invention of the Information Age*. MIT Press.
- Blackler, F., Reed, M., & Whitaker, A. (1993). Editorial introduction: Knowledge workers and contemporary organizations. *Journal of Management Studies*, 30, 851–862.
- Blok, A., & Downey, G. (2003). *Uncovering labour in information revolutions, 1750-2000*. Cambridge University Press.
- Boehm, B.W. (1973). Software and Its Impact: A Quantitative Assessment. *Datamation*, May 1973: 48–59.

- Bosworth, M. (2004). *Vocational Training Programs in Encyclopaedia of Prisons and Correctional Facilities*, Sage Publications.
- Brandon, D.H. (1971). User requirements and in the business and commercial fields. In F. Gruenberger (ed.) *Expanding Use of Computers in the 1970s: Markets, Needs, Technology*, Prentice-Hall.
- Braverman, H. (1998). *Labor and monopoly capital: The degradation of work in the twentieth century*. NYU Press.
- Bresnahan, T. F., & Greenstein, S. (1999). Technological competition and the structure of the computer industry. *The Journal of Industrial Economics*, 47(1), 1–40.
- Bresnahan, T. and Malerba F. (1999), ‘Industrial dynamics and the evolution of firms and nations competitive capabilities in the world computer industry’ in D. Mowery and R. Nelson (eds.), *The Sources of Industrial Leadership*, Cambridge University Press.
- Bricklin, D. & Frankston, B. (2004). *Oral history interview with Dan Bricklin and Bob Frankston*. Charles Babbage Institute.
- Brint, S. (1994). *In an age of experts: The changing role of professionals in politics and public life*. Princeton University Press.
- Brock, G. W. (1975) *The U.S. Computer Industry: A Study of Market Power*. Ballinger
- Brooks, F.P. Jr. (1975). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.
- Brooks, F. P. “No Silver Bullet: Essence and Accidents of Software Engineering.” *IEEE Computer* 20 (4) (1987): 10–19.
- Bryman, A. (2012). *Social Research Methods*. Oxford University Press.
- Bullen, C. V., Abraham, T., Gallagher, K., Simon, J. C., & Zweig, P. (2009). IT Workforce Trends: Implications for Curriculum and Hiring. *Communication of the Association of Information Systems*, 24, 129–140.
- Burawoy, M. (1985). *The politics of production: Factory regimes under capitalism and socialism*. London: Verso.
- Burrows, J.H. (1969) A panel session –Computers and the unprivileged. In *Proceedings of 1969 Spring Joint Computer Conference*, p. 35-14.
- Campbell-Kelly, M. (1980). Programming the Mark I: Early Programming Activity at the University of Manchester. *Annals of the History of Computing*, 2, 130–168.
- Campbell-Kelly, M. (1995). Development and Structure of the International Software Industry, 1950-1990. *Business and Economic History*, 24(2), 73–110.
- Campbell-Kelly, M. (2003), *A History of the Software Industry: From Airline Reservations to Sonic the Hedgehog*, Cambridge, MA: MIT Press.
- Campbell-Kelly, M. (2004). Not All Bad: An Historical Perspective on Software Patents. *Michigan Telecommunication & Technology Law Review*, 11, 191.
- Campbell-Kelly, M. and Aspray. W. (1996) *Computer: A History of the Information Machine*. New York: Basic Books.

- Campbell-Kelly, M., Aspray, W., Ensmenger, N., & Yost, J. R. (2013). *Computer: A History of the Information Machine* (3rd ed.). Boulder: Westview Press.
- Campbell-Kelly, M., & Garcia-Swartz, D. D. (2008). Economic perspectives on the history of the computer time-sharing industry, 1965–1985. *IEEE Annals of the History of Computing*, 1(1), 16–36.
- Campbell-Kelly, M., & Garcia-Swartz, D. D. (2013). The history of the internet: the missing narratives. *Journal of Information Technology*, 28 (1), 18–33.
- Cappelli, P. (1999). *The New Deal at Work: Managing the Market-driven Workforce*. Harvard Business School Press.
- Carroll, P. (1993). *The Unmaking of IBM*. Crown Publishers.
- Casey, C. (1995). *Work, Self and Society: After Industrialism*, Routledge.
- Casner-Lotto, J. (1988) Achieving cost savings and quality through education: IBM's systems approach. In J. Casner-Lotto and Associates, *Successful Training Strategies, Twenty Six Innovative Corporate Models*, Jossey-Bass Publishers.
- Castells, M. (2001). *The Internet galaxy: Reflections on the Internet, business, and society*. Oxford University Press.
- Castells, M. (2010). *The Information Age: Economy, Society, and Culture, Vol. I, The Rise of the Network Society* (2nd ed.). Wiley-Blackwell.
- Ceruzzi, P. (1991). When Computers Were Human. *Annals of the History of Computing*, 13, 237–244.
- Ceruzzi, P. (2003). *A History of Modern Computing*. The MIT Press.
- Christopherson, S. (2004). The Divergent Worlds of New Media: How Policy Shapes Work in the Creative Economy. *Review of Policy Research*, 21(4), 543–558.
- Ciborra, C. (1996). The Platform Organization: Recombining Strategies, Structures, and Surprises. *Organization Science*, 7 (2), pp. 103-118.
- Coha, S. (1982) CDP – 20 years of certification: Portrait of a profession, *Data Management*, Sep. 1982.
- Collins, R. (1979) *Credential Society: An Historical Sociology of Education and Stratification*, Academic Press Inc.
- Coupe, Dick (2013) *There Is Always a Reason to Dance*, iUniverse.
- Cowan, R. S. (1997). *A Social History of American Technology*. Oxford: Oxford University Press.
- Crompton, R. (1990). Professions in the Current Context. *International Sociology*, 4(5), 147–166.
- Cusumano, M. A. & Selby, R. W. (1995). *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. Touchstone.
- Cusumano, M. (2004) *The Business of Software*, Free Press.

- Dahlbom, B. & Mathiassen, L., 1997. The future of our profession. *Communications of the ACM*, 40(6), p.80–89.
- Damarin, A. K. (2006). Rethinking Occupational Structure: The Case of Web Site Production Work. *Work and Occupations*, 33(4), 429–463.
- Damarin, A.K. (2013) ‘The network-organized labor process: control and autonomy in web production work’, *Research in the Sociology of Work*, 24, p. 177-205.
- Davidson, E. J. (2011). ‘Hey professor, why are you teaching this class?’ Reflections on the relevance of IS research for undergraduate students. *European Journal of Information Systems*, 20(2), 133-138.
- Davies, C. (1980). Making sense of the census in Britain and the U.S.A.: The changing occupational classification and the position of nurses. *Sociological Review*, 28 (3) p. 581-609.
- DeLamarter, R.T. (1986) *Big Blue: IBM’s Use and Abuse of Power*. Dodd, Mead.
- Delton, J. A. (2009). *Racial Integration in Corporate America, 1940-1990*. Cambridge University Press.
- DeNelsky, G. Y., & McKee, M. G. (1974). Prediction of Computer Programmer Training and Job Performance Using the AABP Test. *Personnel Psychology*, (27), 129–137.
- Denning, P.J., (2001). When IT becomes a profession. In his (ed.) *The invisible future: The seamless integration of technology in everyday life*, p.295–325. McGraw-Hill.
- Dickmann, Robert A., and John Lockwood. (1966). “1966 Survey of Test Use in Computer Personnel Selection. Technical Report.” In *Proceedings of the Fourth SIGCPR Conference on Computer Personal Research*. New York: ACM Press.
- DiMaggio, P.J. (1991). ‘Constructing an Organizational Field as a Professional Project: U.S. Art Museums, 1920-1940’ in W.W. Powell and P.J. DiMaggio (eds.) *The New Institutionalism in Organizational Analysis*, Chicago: IL: University of Chicago Press.
- DiMaggio, P.J., and Powell, W.W. (1983) The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields. *American Sociological Review* ,Vol. 48, No. 2, pp. 147-160
- Doletta, T.A., Bernstein, M.I., Dickson, R.S. Jr., France, N.A., Rosenblatt, B.A., Smith, D.M., Steel, T.B. Jr., (1976) *Data Processing in 1980-1985: A Study of Potential Limitations to Progress*, Wiley-Interscience.
- Drandell, M. (1990) *IBM: The Other Side: 101 Former Employees Look Back*. Quail Press.
- Dray, W. H. (1989). *On history and philosophers of history*. Brill.
- Dunkerley, D. (1975). *Occupations and Society*, Routledge and Kegan Paul.
- Early, S., and Wilson, R. (1986). “Do unions have a future in high technology?” *Technology Review*, 89 (October): 56–65.
- Edgerton, T. (1999). “Employers as course developers: Are they the new educational



institutions?” In Stacey, N.G. (ed.) *Competence without Credentials*, U.S. Department of Education.

Edwards, R. C., & Edwards, R. (1979). *Contested terrain: The transformation of the workplace in the twentieth century*. Basic Books.

Egan, E. A. (1997). *The Spatial Dynamics of the U.S. Computer Software Industry*, Unpublished PhD Dissertation, University of California, Berkley.

Egan, E. A. (2000). ‘Application districts: An emerging spatial form in the computer software industry’, *Journal of Comparative Policy Analysis: Research and Practice*, 2, 321–344.

Eilbirt, H. (1959). The development of personnel management in the United States. *Business History Review*, 33(03), 345–364.

Einstein, M. E., & Franklin, J. C. (1986). Computer manufacturing enters a new era of growth. *Monthly Labor Review*, 109(9), pp. 9-16.

Ensmenger, N.L., (2001a). The “question of professionalism” in the computer fields. *IEEE Annals of the History of Computing*, p.56–74.

Ensmenger, N.L., (2001b). *From “black art” to industrial discipline: The software crisis and the management of programmers*, Unpublished Ph.D dissertation, University of Pennsylvania.

Ensmenger, N. (2009) From Computer Operators to Operating Systems: The Hidden Costs of Business Computing. Paper presented at the 2009 Social History of Technology Conference.

Ensmenger, N.L., (2010a) *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*, The MIT Press.

Ensmenger, N. L. (2010b). Making Programming Masculine. In T. Misa, *Gender Codes: Women and Men in the Computing Professions*, pp. 115–141. Wiley.

Ensmenger, N. (2012). The Digital Construction of Technology: Rethinking the history of computers in society. *Technology and Culture*, 53(4), 753–776.

Ensmenger, N., and Aspray, W. (2002) Software as a Labor Process. In Ulf Hashagen, Reinhard Keil-Slawik, and Arthur L. Norberg (eds.), *History of Computing: Software Issues*, pp. 139–166. Springer-Verlag.

Ettore, B. (1994). The Contingency Workforce Moves Mainstream. *Management Review* 83(2): 10–16.

Evans, D. S., A. Hagi and R. Schmalensee, (2006) *Invisible Engines: How Software Platforms Drive Innovation and Transform Industries*, The MIT Press, Cambridge, MA:.

Evetts, J. (2011). Sociological Analysis of Professionalism: Past, Present and Future. *Comparative Sociology*, 10(1), 1–37.

Fay, T. C. (1953) *Applied Mathematics as a responsibility of Mathematical Profession*, Proceedings of the Conference on Training in Applied Mathematics, Columbia

University: New York. Pp. 89-90

Ferguson, C. H. & Morris, C. R. (1994) *Computer Wars: The Fall of IBM and the Future of Global Technology*. Times Books.

Filipczak, Bob. (1995). Certifiable. *Training*, 32(8), 38-42.

Fincham, R., Fleck, J., Procter, R. N., Scarbrough, H., Tierney, M., & Williams, R. (1995). *Expertise and innovation: information technology strategies in the financial services sector*. Oxford University Press.

Firth, D., J. King, H. Koch, C.A. Looney, P. Pavlou, and E.M. Trauth (2011). Addressing the Credibility Crisis in IS. *Communication of the Association for Information Systems*, (28), Article 13, pp. 199–212.

Fisher, F. M., McKie, J. W., & Mancke, R. B. (1984). *IBM and the US data processing industry: an economic history*. Praeger Publishers.

Fishman, K. D. (1981) *The Computer Establishment*. Harper & Row.

Flamm, K. (1988). *Creating the Computer: Government, Industry, and High Technology*. Brookings Institution.

Fligstein, N., (2001). *The Architecture of Markets: An Economic Sociology of Twenty-first-century Capitalist Societies*, Princeton University Press.

Fligstein, N. (2008). Myths of the Market. In A. Ebner & N. Beck (eds.), *The Institutions of the Market* (pp. 131–156). Oxford University Press.

Flynn, P.M. (1988) *Facilitating Technological Change: The Human Resource Challenge*, Ballinger Publishing Company.

Flynn, P.M. (1991). The Life-Cycle Model for Managing Technological Change. In P.B. Doeringer and associates, *Turbulence in the American Workplace*, Oxford University Press.

Forrester, J. W. (1951). Digital Computers: Present and Future Trends. In the proceedings of the Joint AIEE-IRE 51' Computer Conference, pp. 109–114.

Frank, W.L. (1979). *The New Software Economics*, United States Professional Development Institute, Inc., Silver Springs, Maryland, 1979.

Freeman, C., & Soete, L. (1994). *Work for all or mass unemployment? Computerised technical change into the twenty-first century*. London: Pinter.

Freidson, E. (1970). *Profession of medicine: a study of the sociology of applied knowledge*. University of Chicago Press.

Freidson, E. (1984). *Doctoring together: A study of professional social control*. Chicago: University of Chicago Press.

Freidson, E. (1988) *Professional Powers: A Study of the Institutionalization of Formal Knowledge*. University of Chicago Press.

Freidson, E. (2001). *Professionalism, the third logic: on the practice of knowledge*. University of Chicago press.

- Friedman, A. L. (1990). Four phases of information technology: Implications for forecasting IT work. *Futures*, 22(8), 787–800.
- Friedman, A. L., & Cornford, D. S. (1989). *Computer systems development: history, organization, and implementation*. John Wiley and Sons.
- Fritz, W. B. (1996). The women of ENIAC. *IEEE Annals of the History of Computing*, 18, 13–28.
- Funk, J.L. (2012) ‘The Unrecognized Connection between Vertical Disintegration and Entrepreneurial Opportunities’, *Long Range Planning*, 45 41-59
- Galagan, P.A. (1990). BM Faces the Future – Again. *Training and Development Journal*; 44 (3) p. 36-40;
- Gallie, Duncan. (1991). Patterns of skill change: upskilling, deskilling or the polarization of skills? *Work, Employment & Society*, 5(3) 319-351.
- Gallivan, M.J., Truex III, D.P., and L. Kvasny, 2004. ‘Changing Patterns in IT Skill Sets 1988–2003: A Content Analysis of Classified Advertising’, *Database for Advances in Information Systems*, vol. 35, no. 3, pp. 64–87.
- Garner, L. (1974). Computer Workers as Professionals. *Science for the People*, 6 (6) November, 28-32.
- Gawer, A. and Cusumano, M.A. (2002) *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*, Harvard Business School Press, Boston: MA.
- Ghazawneh, A., & Henfridsson, O. (2012). Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, 23(2), 173–192.
- Geber, B. (1994) A clean break for education at IBM, *Training*, 31 (2) 33-36.
- George, A. L., & Bennett, A. (2005). *Case Studies and Theory Development in the Social Sciences*. MIT Press.
- Gerstner, Jr. L. V. (2003) *Who Says Elephants Can't Dance: How I Turned Around IBM*. HarperCollinsPublishers.
- Glass, R. (2007) ‘Through a glass, darkly: IS – doom and gloom forecasts?’ *Information Systems Management*, 24, 393-4.
- Golemba, B.E. (1995) *Human Computers: The Women in Aeronautical Research*. Manuscript held at Iowa State University, <http://www.add.lib.iastate.edu/spcl/manuscripts/MS307.html>; Accessed 10 Mar 2015.
- Goles, T., Hawk, S. and Kaiser, K.M., (2008) ‘Information technology workforce skills: The software and IT services provider perspective’, *Information Systems Frontiers*, 10 (2), pp.179–194.
- Gorn, S. (1954). “Planning Universal Semi-automatic Coding.” In *Symposium on Automatic Programming for Digital Computers*, Office of Naval Research, Department of the Navy, Washington, D.C., 13-14 May 1954, edited by U.S. Navy Mathematical Computing Advisory Panel, 74–83. Washington, D.C.: U.S. Dept. of Commerce, Office

of Technical Services.

Grad, B. (2002). A Personal Recollection: IBM's Unbundling of Software and Services. *IEEE Annals of the History of Computing*, 24, 64–71.

Green, F. (2013). *Skills and Skilled Work: An Economic and Social Analysis*. Oxford, UK: Oxford University Press.

Greenbaum, J. (1979). *In the name of efficiency: A study of change in data processing work*. Philadelphia: Temple University Press.

Greenwood, R., Oliver, C., Sahlin, K. and R. Suddaby (2008). 'Introduction', in their (eds.) *The SAGE Handbook of Organizational Institutionalism*, pp. 1-46, London: Sage.

Grier, D. (1996). 'The ENIAC, the Verb "to program" and the Emergence of Digital Computers', *IEEE Annals of the History of Computing*, 18, 51–55.

Grier, D. (2005). *When Computers Were Human*. Oxford: Princeton University Press.

Gutchess, J.F. (1985) *Employment Security in Action: Strategies that Work*, Pergamon Press.

Haigh, T. (2001) Inventing information systems: The systems men and the computer, 1950-1968. *The Business History Review*, 75(1), p.15–61.

Haigh, T. (2010). Computing the American Way: Contextualizing the Early US Computer Industry. *IEEE Annals of the History of Computing*, 32(2), 8–20.

Hall, P. A., & Taylor, R. C. (1996). Political science and the three new institutionalisms. *Political studies*, 44(5), 936-957.

Hall, R. (2010) 'Renewing and revising the engagement between labour process theory and technology', in P. Thompson and C. Smith (eds.) *Working Life: Renewing Labour Process Analysis*, Palgrave Macmillan, et. al. 1985

Halpern, Mark I. "Memoirs (Part 1)." *IEEE Annals of the History of Computing* 13 (1) (1991): 101–111.

Hatch, J. and Clinton, A. (2000) Job growth in the 1990s: a retrospect. *Monthly Labor Review*, Sep, p.3-18.

Head, R. V. (1971) *A Guide to Packaged Systems*. Wiley-Interscience.

Hebden, J.E. (1979) 'Patterns of work identification', *Sociology of Work and Occupations*, 2 (2) p. 107-132.

Holzer, A., & Ondrus, J. (2011). Mobile application market: A developer's perspective. *Telematics and Informatics*, 28(1), 22–31.

Hopper, G. M. (1981). Keynote address. In *History of programming languages, Vol. I*, pp. 7–20. ACM.

Hopper, G. M. (1952/1988). The Education of a Computer. Proceedings of the Association for Computing Machinery Conference, Pittsburgh, Pennsylvania, May 1952. Reprinted in *Annals of the History of Computing*, 9, 271–281.

Howell, M.C. & Prevenier, W., (2001) *From Reliable Sources: An Introduction to*

*Historical Methods*, Cornell University Press.

Hughes, E.C. (1942). The study of institutions. *Social Forces*, 20 (3), pp. 307-310.

Humphrey, W. S. (2002). Software unbundling: a personal perspective. *IEEE Annals of the History of Computing*, 24(1), 59–63.

Iivari, J., Hirschheim, R., & Klein, H. K. (2008). ‘Challenges of professionalization: bridging research and practice through a body of knowledge for IT specialists’ in D. Avison, G. M. Kasper, B. Pernici, I. Ramos, & D. Roode (Eds.), IFIP International Federation for Information Processing, Volume 274; *Advances in Information Systems Research, Education and Practice* (pp. 15–27). Boston: Springer.

Iverson, K. E. (1955) Graduate Instruction and Research, in Jacobson, A. (Ed.) Proceedings of the First Conference on Training Personnel for the Computing Machine Field. Detroit: Wayne State University Press.

Jacobs, J. F. (1983). SAGE overview. *Annals of the History of Computing*, 5(4), 323–329.

Jacobson, A. (Ed.) (1955) Proceedings of the First Conference on Training Personnel for the Computing Machine Field. Detroit: Wayne State University Press.

Jacoby, S. (1985). *Employing bureaucracy : Managers, unions, and the transformation of work in American industry, 1900-1945*. New York: Columbia University Press.

Jacoby, S. (2008). *Employing bureaucracy: Managers, unions, and the transformation of work in American industry, 1900-1945*. Revised Edition. New York: Columbia University Press.

Jarvis, P. (2001). *Universities and corporate universities: The higher learning industry in global society*. London: Kogan Page.

Jesiek, B.K. (2006) ‘The Sociotechnical Boundaries of Hardware and Software: A Humpty Dumpty History’, *Bulletin of Science, Technology & Society*, 26(6), p.497.

Joseph, D., Boh, W. F., Ang, S., & Slaughter, S. (2012). The career paths less (or more) traveled: A sequence analysis of IT career histories, mobility patterns, and career success. *MIS Quarterly*, 36(2), 427–452.

Joseph, D., Ng, K. Y., Koh, C., & Ang, S. (2007). Turnover of information technology professionals: a narrative review, meta-analytic structural equation modeling, and model development. *Mis Quarterly*, 31(3), 547-577.

Kaarst-Brown, M.L. & Guzman, I.R., (2005) ‘Who is the IT workforce?: challenges facing policy makers, educators, management, and research’, in *Proceedings of the 2005 ACM SIGMIS CPR conference on Computer personnel research*. ACM, p. 1–8.

Kalleberg, A. L. (2007). *The Mismatched Worker*. WW Norton & Company.

Kalleberg, A. L. (2011). *Good Jobs, Bad Jobs: The Rise of Polarized and Precarious Employment Systems in the United States, 1970s-2000s*. Russell Sage Foundation.

Kalleberg, A.L. and Berg, I., (1987) *Work and Industry: Structures, Markets and Processes*, Plenum Publishing Corporation, NY.

- Kallinikos, J. (2003). Work, human agency and organizational forms: an anatomy of fragmentation. *Organization Studies*, 24(4), 595–618.
- Kallinikos, J. (2004) Farewell to constructivism: technology and context-embedded action. In C. Avgerou, C. Ciborra, & F. Land, eds. *The Social Study of Information and Communication Technology Innovation Actors and Contexts*. Oxford University Press, pp. 140-161.
- Kallinikos, J. (2011) *Governing through technology: Information artefacts and social practice*. Palgrave Macmillan.
- Kanter, R.M. (1995), 'Nice work if you can get it: the software industry as a model for tomorrow's jobs', *American Prospect*, 23 (Fall), 52–8.
- Kaplan, D., & Lerouge, C. (2007). Managing on the edge of change: human resource management of information technology employees. *Human Resource Management*, 46(3), 325–330.
- Katz, M. B. (1972). Occupational classification in history. *The Journal of Interdisciplinary History*, 3(1), 63-88.
- Kaufman, B. E. (2008). *Managing the human factor: The early years of human resource management in American industry*. Cornell University Press.
- King, J. L., Gurbaxani, V., Kraemer, K. L., McFarlan, F. W., Raman, K. S., & Yap, C. S. (1994). Institutional Factors in Information Technology Innovation. *Information Systems Research*, 5, 139–169.
- King, J. L., & Lyytinen, K. (2004). Reach and grasp. *MIS Quarterly*, 539-551.
- King, J. L., & Lyytinen, K. (Eds.). (2006). *Information systems: The state of the field*. Chichester, UK: John Wiley & Sons.
- Kling, R. (2003). Critical professional education about information and communications technologies and social life. *Information Technology & People*, 16(4), pp.394-418.
- Kling, R. and Gerson, E. (1977). The social dynamics of technical innovation in the Computing World. *Symbolic Interaction* 1,2 (Spring): 24-43.
- Knights, D., & Murray, F. (1994). *Managers divided: Organisation politics and information technology management*. Wiley & Sons.
- Knuth, D.E. (1968), *The Art of Computer programming. Addison-Wesley Series in Computer Science and Information Processing*, Reading, MA: Addison- Wesley.
- Knuth, D. E. and Pardo, L. T. (1977). The Early Development of Programming Languages. Reprinted in *Encyclopedia of Computer Science and Technology*, 7. NY: Marcel Dekker Inc. reprinted in Knuth, D. E. (2002) *Selected Papers in Computer Languages Stanford*, CA: Center for the Study of Language and Information.
- Kochan, T. A., MacDuffie, J. P., & Osterman, P. (1988). Employment security at DEC: Sustaining values amid environmental change. *Human Resource Management*, 27(2), 121–143.
- Kosso, P. (2009) Philosophy of Historiography. In A. Tucker (ed.) *A Companion to the*

- Philosophy of History and Historiography*, John Wiley & Sons.
- Kraft, P. (1977) *Programmers and Managers: The Routinization of Computer Programming in the United States*. Springer-Verlag, New York.
- Kraft, P., & Dubnoff, S. (1986). 'Job content, fragmentation, and control in computer software work', *Industrial Relations*, 25, 184–196.
- Krause, E.A. (1982) *Division of Labor: A Political Perspective*, London, Greenwood Press.
- Krause, E.A. (1996) *Death of the Guilds: Professions, States, and the Advance of Capitalism, 1930 to the Present*, Yale University Press.
- Land, F. (2010). 'The use of history in IS research: An opportunity missed?', *Journal of Information Technology*, 25, 385–394.
- Lane, C. M. (2011) *A Company of One: Insecurity, Independence, and the New World of White-Collar Unemployment*. ILR Press.
- Langlois, R. N. (1992). External economies and economic progress: The case of the microcomputer industry. *Business History Review*, 66(01), 1–50.
- Lanzara, G.F. and Patriotta, G., (2007) 'The Institutionalization of Knowledge in an Automotive Factory: Templates, Inscriptions, and the Problem of Durability' *Organization Studies*, 28(5), pp.635–660.
- Larson, M.S., (1977) *The rise of professionalism: A Sociological Analysis*, University of California Press.
- Larson, M.S. (2013). *The rise of professionalism: A Sociological Analysis*, Revised Edition, Transaction Publishers.
- Lawson, C. (1962). A Survey of Computer Facility Management. *Datamation* 8, no. 7: 29–32.
- Lazonick, W. (2009). *Sustainable Prosperity in the New Economy?: Business Organization and High-tech Employment in the United States*. W.E. Upjohn Institute for Employment Research.
- Leicht, K., & Fennell, M. L. (2001). *Professional Work: A sociological approach*. Malden, MA: Blackwell.
- Leicht, K.T. (2005). 'Professions' in George Ritzer (ed.) *Encyclopedia of social theory*, p. 603–606, Thousand Oaks, CA: Sage.
- Leicht, K.T., and M.L. Fennell (2008). 'Institutionalism and the professions' in Greenwood, R., Oliver, C., Sahlin, K. and R. Suddaby (eds.), *The SAGE Handbook of Organizational Institutionalism*, pp. 431-548, London: Sage.
- Leonardi, P. M. (2012). *Car Crashes without Cars: Lessons about simulation technology and organizational change from automotive design*, Cambridge, MA: MIT Press.
- Levina, N. & Xin, M. (2007). 'Research Note - Comparing IT Workers' Compensation

- Across Country Contexts: Demographic, Human Capital, and Institutional Factors', *Information Systems Research*, 18(2), pp.193–210.
- Levis, K. (2009). *Winners & Losers: Creators & Casualties of the Age of the Internet*. Atlantic Books.
- Light, J. S. (1999). When computers were women. *Technology and culture*, 40, 455–483.
- Looney, C.A., and A.Y. Akbulut (2007) "Combating the IS Enrollment Crisis: The Role of Effective Teachers in Introductory IS Courses", *Communications of the Association for Information Systems*, (19) 38, pp. 781–805.
- Looney, C. A., Firth, D., Koch, H., Cecez-Kecmanovic, D., Hsieh, J., Soh, C., Valacich, J. S. and Whitley, E.A. (2014) The Credibility Crisis in IS: A Global Stakeholder Perspective. *Communications of the Association for Information Systems*: Vol. 34, Article 61.
- Loseby, P. H. (1992) *Employment Security: Balancing Human and Economic Considerations*. Quorum Books.
- Lowell, B. L. (2000). H-1B temporary workers: Estimating the population. *Center for Comparative Immigration Studies*.
- Mahoney, M. S. (2002). Software as Science-Science as Software. In U. Hashagen, R. Keil-Slawik, & A. Norberg (Eds.), *History of Computing: Software Issues* (pp. 25–48). Springer.
- Mahoney, M. S. (2005). The histories of computing(s). *Interdisciplinary Science Reviews*, 30(2), 119–135.
- Markham, E. (1968) EDP Schools: An Inside View, *Datamation* 14, no. 4, 22–27.
- Markus, M.L. (1999) "Thinking the Unthinkable: What Happens If the Is Field as We Know it Goes Away?". In W. Currie and R. D. Galliers (eds.). *Rethinking MIS*, Oxford: Oxford University Press, 1999, pp. 175-203.
- Markusen, A. R. (1983). High-tech jobs, markets and economic development prospects: evidence from California. *Built Environment (1978-)*, 18-28.
- Marwick, A. (2001). *The new nature of history: Knowledge, evidence, language*. Lyceum Books, Incorporated.
- Mason, R.O., McKenney, J.L. & Copeland, D.G. (1997). 'Developing an Historical Tradition in MIS Research', *MIS quarterly*, 21(3).
- McClellan, S.T. (1984). *The Coming Computer Industry Shakeout: Winners, Losers, and Survivors*. John Wiley and Sons.
- McCracken, D.D., Weiss, H., and Lee T. (1966). *Programming Business Computers*. Wiley, New York; Chapman and Hall.
- McCullagh, C. B. (2004). *The Logic of History*. London: Routledge.
- McCullagh, C. B. (1978). 'Colligation and Classification in History', *History and Theory*, 17(3),1–19.



- McNamara, W.J., and Hughes, J.L. (1961). "A Review of Research on the Selection of Computer Programmers." *Personnel Psychology* 14 (1): 39–51.
- McNamara, W.J. (1967) The Selection of Computer Personnel: Past, Present, Future. In *SIGCPR '67: Proceedings of the Fifth SIGCPR Conference on Computer Personnel Research*, 52–56. New York: ACM Press.
- McNerney, D. J. (1996) HR practices: HR adapts to continuous restructuring, *HR Focus*, 73 (6) 1-6.
- Meister, J. C. (1998). *Corporate Universities: Lessons in Building a World-Class Workforce*, McGraw-Hill.
- Messerschmitt, D. G., & Szyperski, C. (2005). *Software Ecosystem: Understanding an Indispensable Technology and Industry*. Cambridge, MA: MIT Press.
- Metropolis, N. C. (1989). The Case Against Automatic Programming. *Annals of the History of Computing*, 11, 322–326.
- Miller, J. (1988) Jobs and work, in N.J. Smelser (ed.) *Handbook of Sociology*, Sage Publications.
- Mills, C. W. (1951). *White Collar, The American Middle Classes*. New York.
- Mills, D. Q. (1985). Planning with people in mind. *Harvard Business Review*, 63 (4): 97-105.
- Mills, D. Q. (1988) *The IBM Lesson: The Profitable Art of Full Employment*. Times Books.
- Misa, T.J. (2012). *Building the Control Data Legacy: The Career of Robert M. Price*, Charles Babbage Institute.
- Misa, T. J. (2013) *Digital State: The Story of Minnesota's Computing Industry*. University of Minnesota Press
- Mitev, N., & De Vaujany, F. X. (2012). 'Seizing the opportunity: towards a historiography of information systems', *Journal of Information Technology*, 27, 110–124.
- Money, J., & Falstrom, D. Z. (2006). Interests and institutions in skilled migration: Comparing flows in the IT and nursing sectors in the US. *Knowledge, Technology & Policy*, 19(3), 44-63.
- Mueller, F., & Dyerson, R. (1999). 'Expert Humans or Expert Organizations?', *Organization Studies*, 20, 225–256.
- Murray, F., & Knights, D. (1990). Inter-managerial competition and capital accumulation: IT specialists, accountants and executive control. *Critical Perspectives on Accounting*, 1(2), 167–189.
- Myers, C., Hall, T., & Pitt, D. (eds.) (1996). *The Responsible Software Engineer*. Springer.
- Nardie, B. (2011). Foreword. In Kallinikos, J. (2011) *Governing through technology: Information artefacts and social practice*. Palgrave Macmillan.

- National Research Council, (2001) *Building a Workforce for the Information Economy*, National Academies Press.
- Nee, V. (2005). 'The New Institutionalisms in Economics and Sociology', in Smelser, N. J., & Swedberg, R. (Eds.) *The Handbook of Economic Sociology*, Second Edition. New Jersey: Princeton University Press.
- Neff, G. (2012) *Venture Labor: Work and the Burden of Risk in Innovative Industries*. MIT press.
- Nelsen, B. J., & Barley, S. R. (1997). "For love or money? Commodification and the construction of an occupational mandate." *Administrative Science Quarterly*, 619–653.
- Norberg, A. L. (2005). *Computers and Commerce*. MIT Press.
- Norman, A. P. (1991). 'Telling it Like it Was: Historical Narratives on Their Own Terms', *History and Theory*, 30, 119–135.
- Norris, W.C. (1986). *Oral history interview with William C. Norris*. Charles Babbage Institute.
- NSF, (1955). *1955 Annual Report*, National Science Foundation.
- NWCET, (2003). *Building a foundation for tomorrow: Skill standards for information technology*. Bellevue, WA. [Northwest Center for Emerging Technologies is now called National Workforce Center for Emerging Technologies.]
- O'Shields, J. (1965) Selection of EDP Personnel. *Personnel Journal*, 44 (9): 472. 51.
- Orden, A. (1967). The emergence of a new profession. *Communications of the ACM*, 10(3), pp.145-147.
- Orlikowski, W.J., (1988) 'The data processing occupation: professionalization or proletarianization?', *Research in the Sociology of Work*, 4, p.95–124.
- Orlikowski, W. J. (1989). Division Among the Ranks: Social Implications of Case Tools (pp. 1–44). MIT Sloan School of Management.
- Orlikowski, W. J, and Baroudi, J. J. (1989) The information systems profession: myth or reality? *Office, Technology and People* 4, no. 1: 13–31.
- Osborn, R. (1965) GE and UNIVAC: Harnessing the High-Speed Computer." *Harvard Business Review* 32 (4): 99–107.
- Osnowitz, D. (2010). *Freelancing Expertise: Contract Professionals in the New Economy*. Ithaca, NY: Cornell University Press.
- Ouchi, W. (1981). Theory Z: How American business can meet the Japanese challenge. *Business Horizons*, 24(6), 82-83.
- Panko, R. R. (2008). IT employment prospects: beyond the dotcom bubble. *European Journal of Information Systems*, 17(3), 182–197.
- Paschell, W. (1958). *Automation and Employment Opportunities for Office Workers: A Report on the Effect of Electronic Computers on Employment of Clerical Workers*, Washington, DC: Bureau of Labor Statistics.

- Paul, R. (2002). "(IS)3: Is information systems an intellectual subject?" *European Journal of Information Systems*, 11, 174–177.
- Perkin, H. (1989). *The Rise of Professional Society*. Routledge
- Perry, D. K. (1962) *Programmer selection procedures: Field Notes*. Santa Monica, Calif.: System Development Corp.,
- Peters, B. (1999). *Institutional theory in political science: The 'new institutionalism'*. London: Pinter.
- Pettigrew, A.M. (1973). Occupational specialization as an emergent process. *Sociological Review*, 21(2), pp.233-278.
- Pierson, P., (2000) 'The limits of design: Explaining institutional origins and change', *Governance*, 13(4), pp.475–499.
- Pierson P. and Skocpol, T. (2002) Historical Institutionalism in Contemporary Political Science, in I. Katznelson and H.V. Milner (eds.) *Political Science: The State of the Discipline*, 371-403. W. W. Norton & Company
- Plunkert, L.M. (1990). The 1980s: A decade of job growth and industry shift. *Monthly Labor Review*, Sep: 3-16.
- Popp, K. M. (2011). Software Industry Business Models. *IEEE Software*, 28(4), 26–30.
- Price, R. M. (2005). *The eye for innovation: recognizing possibilities and managing the creative enterprise*. Yale University Press.
- Pugh, E. W. (1995). *Building IBM: Shaping an Industry and Its Technology*. MIT Press.
- Pugh, E. W., & Aspray, W. (1996). Creating the computer industry. *IEEE Annals of the History of Computing*, 18(2), 7–17.
- Pugh, E. W., Johnson, L. R., & Palmer, J. H. (1991). *IBM's 360 and Early 370 Systems*. The MIT Press.
- Reed, M. I. (1996). Expert Power and Control in Late Modernity: An Empirical Review and Theoretical Synthesis. *Organization Studies*, 17(4), 573–597.
- Rhee, H.A. (1968) *Office Automation in Social Perspective*. Basil Blackwell.
- Riemenschneider, C. K., Armstrong, D. J., & Moore, J. E. (2009). Meeting the demand for IT workers: A call for research. *European Journal of Information Systems*, 18(5), 458–461.
- Roberts, G. (Ed.). (2001). *The History and Narrative Reader*. Routledge.
- Robinson, H.W. (1988). *Oral history interview with Herbert W. Robinson*. Charles Babbage Institute.
- Rogers, E. M. & Larsen J. K. (1984) *Silicon Valley Fever: Growth of High-Technology Culture*. Basic Books
- Rose, M. (2002). IT professionals and organisational ascendancy: theory and empirical critique. *New Technology, Work and Employment*, 17(3), pp.154-169.
- Ross, A. (2004). *No-collar: The humane workplace and its hidden costs*. Temple

University Press.

Rowan, T. C. (1957). Psychological Tests and Selection of Computer Programmers. *Journal of the ACM*, 4 (3) 348–353.

Rowan, T. C. (1958) “The Recruiting and Training of Programmers.” *Datamation* 4 (3) 16–18.

Rowlinson, M. (2004). ‘Historical Analysis of Company Documents’, in C. Cassell & G. Symon (Eds.), *Essential Guide to Qualitative Methods in Organizational Research*. Sage Publications.

Rudy, P. (2004). Justice for Janitors, “Not Compensation for Custodians.” In R. Milkman and V. Ross (eds.) *Rebuilding labor: Organizing and organizers in the new union movement*, 133-49. Cornell University Press.

Rueschemeyer, D. (1983) Professional autonomy and the social control of expertise, in R. Dingwall and P. Lewis (eds.) *The Sociology of the Professions: Lawyers, Doctors and Others*, p. 38-58. London: Macmillan.

Rueschemeyer, D. (1986) *Power and the Division of Labour*, Stanford University Press.

Sammet, J.E. (1991) Programming Languages History. *Annals of the History of Computing* 13 (1) 49.

Saxenian, A. (1994). *Regional advantage*. Harvard University Press.

Sayer, A. (2000). *Realism and Social Science*, London: SAGE Publications.

Scarborough, H. (1995). Blackboxes, Hostages and Prisoners. *Organization Studies*, 16(6), 991–1019.

Schachter, Oscar. (2004). *Oral history interview with Oscar Schachter*. Charles Babbage Institute.

Scott, W. R. (2008). Lords of the Dance: Professionals as Institutional Agents. *Organization Studies*, 29(2), 219–238.

Scott, W.R. (2010) ‘Entrepreneurs and Professionals: The mediating role of institutions’, *Research in the Sociology of Work*, 21, pp. 27–49.

Scott, W.R. (2014). *Institutions and Organizations: Ideas, interests and identities*, London, Sage.

Scott, W.R., Ruef, M., Mendel, P. J., & Caronna, C. A. (2000). *Institutional Change and Healthcare Organizations: From Professional Dominance to Managed Care*. University of Chicago Press.

Seiler, J. (1967, June). Survey of validation studies on computer personnel selection instruments. In *Proceedings of the fifth SIGCPR conference on Computer personnel research* (pp. 43-51). ACM.

Sennett, R. (1999) *The Corrosion of Character: Personal Consequences of Work in the New Capitalism*, W. W. Norton & Company.

Shank, S.E. and Getz, P.M. (1986). Employment and unemployment: developments in 1985. *Monthly Labor Review*, Feb:3-12.

- Shay, J.E. (2012) *Bygone Binghamton: Remembering People and Places of the Past*, AuthorHouse.
- Siegrist, H. (2002). Professionalization/professions in history. *International encyclopedia of the social and behavioral sciences*, 12154-12160.
- Silverman, D. (2013). *Doing Qualitative Research: A Practical Handbook*. SAGE Publications.
- Simpson, R. L. (1985). Social control of occupations and work. *Annual Review of Sociology*, 415–436.
- Sine, W.D. and David, R.J. (2010) ‘Institutions and Entrepreneurship’, *Research in the Sociology of Work*, 21, pp. 1–26.
- Slocum, W.L. (1974). *Occupational Careers: A Sociological Perspective*. Chicago, Aldine Publishing Company.
- Smith, V. (1997). New forms of work organization. *Annual Review of Sociology*, 23, 315-339.
- Somers, M.J. (2010). Using the theory of the professions to understand the IS identity crisis. *European Journal of Information Systems*.
- Spenner, K. I. (1995). Technological Change, Skill Requirements and Education: the Case for Uncertainty. In D. B. Bills (Ed.), *The New Modern Times: Factors Reshaping the World of Work*. State University of New York Press.
- Stark, D., & Girard, M. (2009). Creative Friction in a New-Media Start-Up. In D. Stark, *Sense of Dissonance: Accounts of Worth in Economic Life*, pp. 81–117. Princeton University Press.
- Steinmo, Sven. (2008). Historical Institutionalism. In Donatella Della Porta and Michael Keating (eds.) *Approaches and Methodologies in the Social Sciences: A Pluralist Perspective*, New York: Cambridge University Press, 118-138.
- Steinmueller, W. E. (1996). ‘U.S. Software Industry: an Analysis and Interpretive History’, in D. C. Mowery (Ed.), *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*. Oxford University Press.
- Strauss, G. (1963) ‘Professionalism and occupational associations’, *Industrial Relations*, 2 (2) 7-31.
- Streek, W. and Thelen, K. (2005) ‘Introduction: Institutional Change in Advanced Political Economies’, in their (Eds.) *Beyond Continuity: Institutional Change in Advanced Political Economies*. Oxford: Oxford University Press.
- Swanson, E.B. & Ramiller, N.C., (2004) ‘Innovating mindfully with information technology’, *MIS Quarterly*, pp. 553–583.
- Swanson, E.B. and Ramiller, N.C., (1997) ‘The organizing vision in information systems innovation’, *Organization Science*, 8 (5), pp.458–474.
- Sweet, S. & Meiksins, P. (2008). *Changing Contours of Work: Jobs and Opportunities in the New Economy: Jobs and Opportunities in the New Economy*. Pine Forge Press.

- Swenson, P. (2002). *Capitalists against markets: The making of labor markets and welfare states in the United States and Sweden*. Oxford University Press.
- Thelen, K. (2002). The political economy of business and labor in the developed democracies. in I. Katznelson and H.V. Milner (eds.) *Political Science: The State of the Discipline*, 371-403. W. W. Norton & Company.
- Thelen, K. (2004) *How institutions evolve: The political economy of skills in Germany, Britain, the United States, and Japan*, Cambridge University Press.
- Thelen, K. (2008) Skill formation and Training, in G. Jones and J. Zeitlin (Eds.). (2008). *The Oxford handbook of business history*, p.558-581. Oxford University Press.
- Thielen, D. (1999) *The 12 Simple Secrets of Microsoft Management: How to Think and Act Like a Microsoft Manager and Take Your Company to the Top*. McGraw-Hill.
- Thornton, P. H. (2004). *Markets from Culture: Institutional Logics and Organizational Decisions in Higher Education Publishing*. Stanford, CA: Stanford University Press.
- Thornton, P. H., Ocasio, W., & Lounsbury, M. (2012). *The Institutional Logics Perspective: A New Approach to Culture, Structure, and Process*. Oxford: OUP.
- Tilly, Chris, and Tilly, Charles (1998). *Work under capitalism*. Westview Press.
- Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Research Commentary—Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. *Information Systems Research*, 21(4), 675–687
- Todd, P. A., McKeen, J. D., & Gallupe, R. B. (1995). The evolution of IS job skills: a content analysis of IS job advertisements from 1970 to 1990. *MIS quarterly*, 1-27.
- Torrisi, S. (1998). *Industrial organisation and innovation: an international study of the software industry*. Edward Elgar Publishing.
- Trauth, E. M. & Hafner, C. D. (2000). Meeting the IT skills crisis: An interdisciplinary response. *AMCIS 2000 Proceedings*, 314.
- Tropp, H. S. (1983). A Perspective on SAGE: Discussion. *Annals of the History of Computing*, 5(4), 375–398.
- Tukey, J. (1958). The Teaching of Concrete Mathematics. *American Mathematical Monthly*, 65 (1), 1–9.
- Turner, C. and Hodge, N. (1970) Occupations and professions, in J.A. Jackson (ed.) *Professions and Professionalization*. Cambridge University Press.
- Univac Conference (1990). *Oral History on UNIVAC Conference*. Charles Babbage Institute.
- Usdansky, M. L. & Espenshade, T. J. (2000). The H-1B Visa Debate in Historical Perspective: The Evolution of US Policy Toward Foreign-Born Workers. *Center for Comparative Immigration Studies*.
- van Jaarsveld, D. (2004). Collective Representation Among High-Tech Workers at Microsoft and Beyond: Lessons from WashTech/CWA. *Industrial Relations: a Journal of Economy and Society*, 43(2), 364–385.

- van Jaarsveld, D. (2007). Boom and bust: lessons from the Information Technology workforce. In J. Amman, T. Carpenter, & G. Neff, (eds.) *Surviving the New Economy*. London: Paradigm Publishers, pp. 119-132.
- Van Maanen, J. & Barley, S. R. (1984). Occupational communities: Culture and control in organizations. *Research in Organizational Behavior*. Vol 6, 1984, 287-365.
- Volti, R. (2007). *An Introduction to the Sociology of Work and Occupations*. Pine Forge Press.
- Vorhaus, L.C. (2000) *Memoirs of a Middle Child*, Xlibris Corporation
- Walsh, W.H. (1958) *An Introduction to Philosophy of History*. Hutchinson, London.
- Walsham, G. (1993). *Interpreting Information Systems in Organizations*. John Wiley & Sons.
- Walsham, G. (1996). Ethical theory, codes of ethics and IS practice. *Information Systems Journal*, 6(1), p.69–81.
- Wang, P. & Ramiller, N. C. (2009). Community learning in information technology innovation. *MIS quarterly*, 709-734.
- Wang, P. & Swanson, E.B., (2007) ‘Launching professional services automation: Institutional entrepreneurship for information technology innovations’, *Information and Organization*, 17 (2), pp. 59–88.
- Watson, Jr. T. J. & Petre, P. (1990) *Father, Son & Co.: My Life at IBM and Beyond*. Bantam Books.
- Webster, F. (2006). *Theories of the Information Society*. Routledge.
- Werle, R. (2012). ‘Institutions and Systems: Analyzing technical innovation processes from an institutional perspective’, in Bauer, J., Lang, A. and V. Schneider (eds.) *Innovation Policy and Governance in High-Tech Industries*, Berlin, Springer.
- White, H. (1992). ‘Historical Representation and the Problem of Truth’, in S. Friedlander (Ed.), *Probing the Limits of Representation*. Cambridge, MA: Harvard University Press.
- Whyte, W.H. (1956). *The Organization Man*, Simon & Schuster.
- Wilensky, H.L. (1964). “Professionalization of everyone?” *American Journal of Sociology*, 70 :137-158.
- Willoughby, T. C. (1975). Psychometric characteristics of the CDP examination. In *Proceedings of the thirteenth annual SIGCPR conference* (pp. 152-160). ACM.
- Wilson, G., & Blain, M. (2001). Organizing in the New Economy. *WorkingUSA*, 5(2), 32-58.
- Wyly, S. (2002). *Oral history interview with Sam Wyly*. Charles Babbage Institute.
- Wyly, S. (2008). *1,000 Dollars and an Idea*. Newmarket Press.
- Xia, R., Rost, M., & Holmquist, L. E. (2010). *Business Models in the Mobile Ecosystem*, Presented at the Ninth International Conference on Mobile Business.

- Yates, JoAnne. (1995) "Application Software for Insurance in the 1960s and Early 1970s." *Business And Economic History* 24 (1): 123–134.
- Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Information Systems Research*, 21, 724–735.
- Yost, J.R. (2005). *The Computer Industry*, Greenwood Press.
- Zabusky, S. E. (1997) Computers, clients, and expertise: negotiating technical identities in a nontechnical world. *Between Craft and Science, Technical Work in US Settings*
- Zald, M. N. (2002). Spinning Disciplines: Critical Management Studies in the Context of the Transformation of Management Education. *Organization*, 9, 365–385.
- Zhang, X., Ryan, S. D., Prybutok, V. R., & Kappelman, L. (2012). Perceived obsolescence, organizational embeddedness, and turnover of it workers: an empirical study. *SIGMIS Database*, 43.
- Ziob, L. (2000) Time is flying; 10 years of IT certification. *Certification Magazine*.  
[https://web.archive.org/web/20030910013917/http://www.certmag.com/issues/feb00/feature\\_ziob.cfm](https://web.archive.org/web/20030910013917/http://www.certmag.com/issues/feb00/feature_ziob.cfm); Accessed 10 Mar 2015.
- Zucker, L. G. (1987). Institutional Theories of Organization. *Annual Review of Sociology*, 13, 443–464.