Software ENgineering

History of the software industry: the challenge

G. Alberts, A. van den Bogaard, M. Campbell-Kelly, F. Veraart

Note SEN-R0511 June 2005

ISSN 1386-369X

History of the software industry: the challenge

ABSTRACT
Abstract Martin Campbell-Kelly opened a new field in the history of computing in his groundbreaking From airline reservation to Sonic the Hedgehog; a history of the software industry. The book is discussed by Adrienne van den Bogaard and Frank Veraart and by Gerard Alberts, followed by a reply by the author. Van den Bogaard and Veraart join great appreciation for the three-sector division Campbell-Kelly developed to describe the history of the software industry, to a slight criticism of his ad hoc-argumentation in explaining why in each sector some enterprises survive and others do not. Lacking, in their view, is a discussion of the dynamics of software itself in the context of emerging practices and businesses. Alberts overcomes his prima facie unease with the "entrepreneurial briefs" of the multitude of software houses, by recognizing the true historian at work. ICT has brought about many changes, including structural changes in the economy. While Campbell-Kelly emphasizes novel aspects of the software economy, in particular the economy of increasing returns, Alberts proposes a more radical interpretation of these phenomena, asking if one call still talk of an "industry". Have we not entered the postindustrial era? In reply, Martin Campbell-Kelly takes the blame for his choices of style, while insisting on his taxonomy of the software industry, and on calling it industry. The present report results from a debate in the Colloquium History of Computing, held at the CWI, June 10-11, 2004.

2000 Mathematics Subject Classification: 01A85
1998 ACM Computing Classification System: K2, K1, D0, J1
Keywords and Phrases: History of Computing; History of Software; History of Technology; Business History; Software industry; Software Entrepreneurs; Software Contracting; Programming Services; Software Products; Mass Market Software Products; "Microsoft economics"; Review Essay
Note: Colloquium History of Computing, CWI, June 10-11, 2004, supported by National Organization for Scientific Research (NWO-exacte wetenschappen, incidentele steun 2004); work carried out as part of CWI theme SEN0, Biography of Aad van Wijngaarden

History of the software industry: the challenge

Gerard Alberts*[a][b]
Adrienne van den Bogaard**
Martin Campbell-Kelly***
Frank Veraart****
*Center for Mathematics and Computer Science (CWI)
and Radboud Universiteit Nijmegen
** Technische Universiteit Delft
***Warwick University
****Technische Universiteit Eindhoven

Abstract

Martin Campbell-Kelly opened a new field in the history of computing in his groundbreaking *From airline reservation to Sonic the Hedgehog; a history of the software industry*. The book is discussed by Adrienne van den Bogaard and Frank Veraart and by Gerard Alberts, followed by a reply by the author.

Van den Bogaard and Veraart join great appreciation for the three-sector division Campbell-Kelly developed to describe the history of the software industry, to a slight criticism of his ad hoc-argumentation in explaining why in each sector some enterprises survive and others do not. Lacking, in their view, is a discussion of the dynamics of software itself in the context of emerging practices and businesses.

Alberts overcomes his prima facie unease with the "entrepreneurial briefs" of the multitude of software houses, by recognizing the true historian at work. ICT has brought about many changes, including structural changes in the economy. While Campbell-Kelly emphasizes novel aspects of the software economy, in particular the economy of increasing returns, Alberts proposes a more radical interpretation of these phenomena, asking if one call still talk of an "industry". Have we not entered the postindustrial era?

In reply, Martin Campbell-Kelly takes the blame for his choices of style, while insisting on his taxonomy of the software industry, and on calling it industry.

The present report results from a debate in the Colloquium History of Computing, held at the CWI, June 10-11, 2004[♪]

1998 ACM Computing Classification System:
K.2. History of Computing---Software
K.1. Computer Industry---Markets
D.0 [Software] General
J. Applications: J.1. Administrative Data Processing---Business

Additional Keywords and Phrases: History of Computing; History of Software; History of Technology; Business History; Software industry; Software Entrepreneurs; Software Contracting; Programming Services; Software Products; Mass Market Software Products; "Microsoft economics"; Review Essay

[♪]Colloquium History of Computing, CWI, June 10-11, 2004, supported by National Organization for Scientific Research (NWO-exacte wetenschappen, incidentele steun 2004)
[a] Work carried out as part of CWI theme SEN-0, Biography of Aad van Wijngaarden
[b] Corresponding author: G.Alberts@cwi.nl

The history of software: new challenges to the historian of technology

review of
Martin Campbell-Kelly *From Airline Reservations to Sonic the Hedgehog. A History of the Software Industry*

by Adrienne van den Bogaard and Frank Veraart

The history of information technology has largely been dominated by looking at the development of hardware and its inventors and producers. Information technology was equated by its electronics, its electronic relays, its vacuum tubes, and circuitry. History of software as a technology has only recently been recognized as a domain of research. Of course, some software pioneers have been looking back on their past, for example discussing the history of programming languages. But the history of software as a technology written by historical experts is a rather new phenomenon.
Campbell-Kelly dared to take up the challenge: he wrote a book on the history of the software industry. Those who think this to be an easy task, are clearly mistaken. As Campbell-Kelly writes, because of the emergence of the personal computer as an everyday tool, people tend to think about Microsoft as the only important producer of software. However, Microsoft has only 10 % of the total software market and acquired this market share only since its successful introduction of the MS-Office package, less than ten years ago. This means that writing the history of the software industry implies much more complex research: it involves many companies and many markets in the context of changing hardware and software technologies, in a rapidly changing society. And unfortunately, good economic and business figures are not available to the historian of this sector.
Campbell-Kelly's book starts with a very nice and neat introductory chapter about the long-term history of the software industry. He introduces a division of the software industry in three sectors: A. the software contractors (since the mid 1950s) B. producers of corporate software products (since the mid 1960s) and C. the mass-market software products (since the late 1970s). Campbell-Kelly explains the rise of these three sectors as opportunities created by changing hardware: the mainframe computers, the launch of the IBM 360 and the emergence of the personal computer. Within one sector companies compete against each other; among the sectors there is hardly any competition—they are more or less mutually exclusive because of different markets and capabilities. And finally, all sectors are dominated by US businesses.
Each of these sectors has its own business characteristics. Software contractors (like Systems Development Corporation, Computer Sciences Corporation) compete on three criteria: economies of scope, cost estimation and project management. For producers of corporate software products (Applied Data Research, Informatics, SAP) different criteria determine who wins and loses:

economies of scale, corporate marketing skills, quality assurance, and pre- and after sales support. And finally for the personal computing mass-market software products (Microsoft, Symantec), economies of scale, marketing capabilities, and user friendliness determine the outcome of the battle.
The book follows this sectorization and periodization. For every sector the author spends one chapter on its origins, and one or two chapters on the final shaping and maturing of the sector. One chapter is fully devoted to the emergence of home and recreational software ("games") and his book ends with reflections on the success of the US software industry.
Campbell-Kelly's book is impressive because of the aggregate view the reader gets from the software industry. After reading the book one has a clear overview of the long-term history of the software sector. The book is less impressive when it comes down to the individual histories of companies. In the field of history of technology it has become common practice to analyse the working of a technology by looking at its origins. One might say, the historian looks at all the variations at one point in time and explains why one of those has been selected by the "selection environment" that consists of a plurality of actors and techniques. The phase of explanation why one technology wins, and the other fails needs consideration: it is never simply a story of the one best technology that wins, nor simply a story of best marketing. Most of the time, a complex story in which technological, social, cultural, political factors and actors all play their roles, is required to explain why some win, and others loose.
The structure of the book – one chapter on origins and one or two chapters on the final shaping of each sector – suggests that Campbell-Kelly indeed adopted such a perspective. However, his explanation of why one company survives while (hundreds of) others fail, suffer from ad-hoc argumentation. At one occasion the genius of a company's founder explains why that company became successful, at another the social intelligence of a company's founder is the determining factor, at yet another occasion the technology is superior to competitors' technologies, and in some cases it is the advantage of the first mover, etcetera. Furthermore, these arguments seem to be often based on actors oral or written judgments. The companies are not mutually compared with a coherent set of variables, and that makes the comparisons ad hoc.
For example, in chapter two, on the origins of the software contractors, the early development of programming languages in the second half of the 1950s, for example by John Backus at IBM, is described as a crucial condition for the emergence of the software sector at all. After its release in the spring of 1957, FORTRAN became the standard language for scientific and engineering applications programming because "it was simply the first efficient and reliable programming language"(page 35). IBM had a good and very skilled programming staff. FORTRAN II, released in 1959, had 50.000 lines of code and had taken 50 programmer years to develop. While IBM developed FORTRAN, the System Development Corporation (SDC) emerged to develop the "software" (which is an anachronistic term for that period as Campbell Kelly rightly argues at page 4) for the Semi-Automatic Ground Environment project (SAGE). IBM had declined this

enormous job because, according to Campbell-Kelly, quoting an IBM manager "IBM didn't see how to employ thousands of programmers when the project would be over"(page 38).
However, further on it turns out (page 47) that SDC developed its own computer language JOVIAL specifically suited for real-time military applications. So, a different hypothesis may be that IBM didn't see the military market for real-time computing as its market (of course, IBM had dominated the market of office machines), and its own emerging capabilities on FORTRAN were not suitable for this market. Deliberate strategic thinking by IBM may have been another reason why IBM had declined the software job for SAGE. This brings us back to the question about the programming languages: in what sense does the explanation about the winner FORTRAN as the first efficient language hold? What is an efficient language? How did this categorization of programming languages emerge? What kind of competition and rivalry has there been among computer languages that has been left out of the story here?
In the last section of chapter two Campbell-Kelly analyses the Software Contracting Startups. The reader gets heroic stories about bright, entrepreneurial, and technical individuals who only needed a pencil to write their programs. The one was better technically, the other managerially or socially, but what they had in common were their heroic efforts and big successes. This scheme of explanation comes back more often in other chapters (for example chapter 3, when the dominance of Advanced Computer Techniques, Applied Data Research and Informatics out of 50-60 starting companies is described) and is really not very convincing. Following this line of reasoning, Campbell Kelly misses the opportunity of mentioning failed software products, strategies and companies. Stories of failed technologies can give extra insight in why others succeeded. For example, Campbell Kelly might have chosen to discuss the failed strategy in attacking the IBM PC standard by Japanese and European companies with the development of MSX computers and its software.

The history of the three software sectors has been analysed by Campbell-Kelly in terms of their market structures. Who are the producers, what are their markets, and how do they compete? The conclusion that the current software industry in fact consists of three sectors, of which each dominating business has its own comparative advantage, is the result of looking at the branch from a historical perspective: by seeing the changing hardware through time, one can see different types of software emerging. However, one doesn't always get a clear picture of the situation at a certain moment in time. For example, if Campbell-Kelly writes about a vast majority of civilian computers used for mundane batch applications in the 1950s (page 48) we start wondering about the sector at all. How many computers were there actually at that time? In the Netherlands, for example, there were only 29 computers in 1959. Of course the US were far ahead of the Dutch, but the point remains: how many organisations did actually have a computer in the US at that time, what did they want to do with it, and how did that affect the opportunities of the software sector, and how did this situation change through time?

The software industry developed in relation to changing hardware, but not only that: the software itself changed. The content of programming, the content of the work that needed to be done, and the problems that occurred have changed over time. Knowledge and skills changed, as well as the required competence. The SDC developed from "programming activities" (chapter 2) to "system architecture" (chapter 3) but this is hardly explained. The practices in which software was developed and applied are not very clearly described. With exceptions here and there, like page 68 when the waterfall method is described, the content of software remains largely black-boxed. For example, in chapters 4 and 5, the development of "software packaging" into "software products" implied new challenges for software. Buyers of the software products should be able to trust the working of the software. Failures and bugs were not allowed anymore. This must have led to new methodologies for producing software, but these are not discussed, not even stated as a question. At several places in the book, new challenges to software seem to appear out of the blue, like multitasking in chapter 8, and also the solutions "multitasking needed to be addressed by a windows operating system" are given and clear to everybody. From a history of technology perspective, we would like to know who actually defined this multitasking problem and for what purpose? And in what sense was the solution clear? In general, we may say that the business historian Campbell-Kelly offers the historians of technology new research questions: what has been the dynamics of software itself in the context of emerging practices and businesses?

Computers have changed society and in fact, it has mostly been the software that did it. That explains (among other things) the relevance of this book. Tempting citations are given about the impact software products in businesses. "*Today, SAP's R/3 runs the back office of half of the world's 500 top companies"*(page197). A thorough examination of this issue is lacking however, which is a pity.

Chapters 7 and 8 are about software for personal computers, Campbell-Kelly makes a big effort to attack modern myths and misconceptions of the personal computing software market. He argues for example that the 'killer application' hypothesis doesn't hold (a big market for personal computing software would also have emerged without the classic VisiCalc spreadsheet), and he argues more than once that Microsoft only dominates the PC software market. Chapter 8 successfully shows that the Microsoft dominance was the outcome of a process of strategic development using the Apple system to test the *Office* programs, luck in using the right economic strategies and persistence in re-releasing *Windows* three times. However we feel ambivalent about how to read Campbell Kelly's strategies to compete or coexist with Microsoft. Are these policy advises, given by the historian who simultaneously thinks that 'the lessons of history' are often exaggerated? (p.303)

And last but not least, what challenges did the software industry face that came from societal and industrial worries, like "privacy" problems, or illegal copying of software, and how did that influence the course of its history?

To conclude – Campbell-Kelly has done a tremendous job to bring together into one picture a whole branch of industry. But it is very much a business history that leaves us historians of technology with a lot of questions. However, the book can certainly function as a steppingstone for further research on the history of software.

But is it industry?

review of
Martin Campbell-Kelly, *From Airline Reservation to Sonic the Hedgehog; A History of the Software Industry*

by Gerard Alberts

Martin Campbell-Kelly has worked something of a miracle, not only breaking new ground and considerably enlarging the field of the history of computing, but also cultivating his new garden so well that he delivers a synthesis at the first harvest. We do get a first comprising history of software entrepreneurship and that is very good value for money. I am inclined to call it a history of entrepreneurship rather than of industry, for two reasons. First, the book offers an endless series of entrepreneurial briefs, so many short stories that the reader is almost certain to get lost at first confrontation. The book not only deserves, it also demands to be read twice. Second, in so many respects does the economic production of software diverge from traditional *industry*, that one may well wonder if it is fair to call it by that name. The shear lack of figures, which strikes the author in his attempt to describe it as industry, is but the most superficial respect in which software production diverges. Economic historians might find a reason here to think twice.
I will elaborate on these two reasons in the first and second section of this review. In a third and final section I will try an appreciation of the questions this books raises to the history of computing. One may criticize Campbell-Kelly for aspects overlooked or issues not raised, but let us rather accept the results of this surprisingly coherent reconnaissance tour and see what it adds to our agenda for historical research. The book offers in three main parts the economic appearance of software in the late 1950s, its evolution in the following decades into product, and into mass market item. This typology and the further qualifications deriving from the economic history present as many challenges to the history of computing. Can we come up with an historical notion of “software”, both in terms history of computing and history of technology, in accordance or conflict with these findings?

1. Entrepreneurial briefs

Overcoming first impressions

Campbell-Kelly wants to be nice to every one of his protagonists, or so it seems. There is a superlative for every entrepreneur. The fastest grower within a sector, the first initial public offering after a long time, et cetera. Every single one gets his mark of uniqueness.
After the general announcement that, “The trajectories of most of the succesful early software startups were similar. They were established by entrepreneurially minded individuals from the technical computing community who, individually or

severally, combined the skills of the technical expert and the business promotor" (p. 50) ---after this general statement--- it starts raining men.
"CSC was founded in Los Angeles in 1959 [...] The promotor was the charismatic Fletcher, then not quite 30 [...] Patrick, another socially adept individual [...] Nutt, the indispensable technical genius [...] An 'introverted mathematician,' he had already achieved mythical status [...]"(p. 52-53) . "CSC [...] had developed project management and cost-estimating skills in compiler development that no other firm possessed [...]" (p. 54)
"Informatics [...] whose growth and prominence owed much to the personality of its principal founder. Walter F. Bauer [...]. Informatics [...] made its mark on the history of software in 1967 by developing Mark IV, which would be the world's best-selling independent software product for 15 years." (p. 58)
"By far the biggest independent player in processing services was Automatic Data Processing (ADP), established in 1949 by the entrepreneur Henry Taub [...]" (p. 62)
"no firm made such a succes of the concept as a core strategy as Perot's [EDS]". (p. 63)
"By 1975, Whitlow Computer Systems was able to advertise that SyncSort was used by more than half of the *Fortune*'s top 50 companies" (p. 102)
"[...] PANVALET in fact went on to achieve sales of $100 million, and Pansophic was regularly among the top 10 independent software firms." (p. 103)
Let us stop here before we meet with Larry Ellison (p. 185). This having so many of the top 10, top 100, or top 500 among its clients is a returning –and clearly not far fetched– criterion among the superlatives in Campbell-Kelly's narrative. We get no time to breathe between two instances of realisation that this is a true business history: "history of the software industry."
In chapter 5 on 'the shaping of the software products industry, the 1970s' the height of chaos is reached in presenting the entrepreneurial studs. And just when one is about ride them all to the stable for a calm winter – and read the rest of the book in spring, perhaps– one realises not properly having read the small letters.
Behind every next chapter, behind every next section, into which the software industry has been broken up, there is a proper rationale to treat it seperately. Moreover, in the absence of standard figures and surveys Campbell-Kelly explains why he picked the examples he did, in his sustained effort to properly represent the sector of the software industry under consideration. Every such sector is treated as a tiny universe of its own. The author explains why a certain firm epitomizes or otherwise adequately represents this universe ---the reason occasionaly being that the available data simply left no alternative. He offers these explanations so succinctly that the reader is tempted to overlook the notice of caution they convey.
For every section there is a well-considered choice of protagonists. Upon realising this feat, my appreciation changed dramatically, from the impression of reading a fetishist of surveys and figures, towards being impressed by the considerate choice of exemplars. Needless to say, the historian of the exemplar is quite a different type of scholar from the historian of the survey. The first takes the

individual case as representative on the basis of judgement on content; the latter will take the average cases. Campbell-Kelly, by adjusting his treatment of all these small universes to the character of the field and to the available data, builds up quite heterogenous chapters. It does not make leisurely reading.

Taxonomy
Nevertheless, it is beyond doubt that the "classification or taxonomy" (p. 3) the book offers, will prove its most lasting value. The author threatens the reader with a classification along three vectors, periodization, economic sector and market. In all three dimensions, i.e. along all three vectors, he proposes a division in three; that would make 27 boxes. Luckily the threat is not exerted: the divisions under the various aspects coincide. The three main sectors within the software market, viz. software contracting, corporate software products, and mass market software products, had the historical benevolence to emerge neatly with intervals of a decade and to address distinct markets. By consequence the book has three main parts each built up of two or three chapters, one to sketch the origin of the sector, and another two to present its shaping and maturing. Chapter 10 'Reflections on the Success of the US Software Industry', is somewhat different in style and purport; more of a critical contribution to innovation studies than to historiography.
The 'Origins of the Software Contractor, the 1950s' in Chapter 2 offers a rather linear story, from IBM's Technical Computing Bureau, its user organization and programming language development, through RAND's spinoff SDC serving as a "university of programmers" working on SAGE, and IBM working on SABRE, to the early startup firms in contracting CUC, CSC, C-E-I-R, and CAI. In the next chapter 3 'Programming Services, the 1960s' the taxonomy really grows branches. Reluctantly following the economic analysis of the day, Campbell-Kelly takes CSSI as the unit of consideration: Computer Software and Services Industry, comprising programming services, processing services, facilities management, and teleprocessing. I tend to disagree with Campbell-Kelly's negative stance towards "conflating programming services with three other activities". As Thomas Haigh ('Software in the 1960s as Concept, Service, and Product', Thomas Haigh. *IEEE Annals of the History of Computing*, January-March **2002**, 5-13) brought to our attention, the notion of software was not as clearly delineated and in fact had a different focus than it has today. In particular consultancy and service could well be included. Hence, taking CSSI for software might convey more historical truth than we are inclined to accept with hindsight.
Distinguishing software products (chapter 4-6) from software contracting is the most basic part of the taxonomy, and the most important one. The gradual transition, 1965 – 1970, from software package to software product, is exemplified by Informatics' Mark IV file management system. Had Mark III and its predecessors been packages, typically customized for each client, Mark IV came as a selfcontained product for the market: to be used without adaptation, with manuals for the user and at a high price (p. 103-109). This is a subtle analysis

providing the context to understand the relative importance of IBM’s unbundling decision, the “crucial inflection point” quoting JoAnne Yates (p. 114).

Reappraising chapter 5

The venerable chapter 5 on ‘The Shaping of the Software Products Industry, the 1970s’, derives its taxonomy, like most other parts of the book, from contemporary sources dividing software products into *systems software* (with a subdivision into five catogories) and *applications* (subdivided into ‘Industry Specific’ and ‘Cross-Industry’). Of course, systems software is operating systems, teleprocessing monitors (equally dominated by IBM), programming aids, utilities, but also database management systems. Surprisingly, database management systems did not count as applications, but as systems software. Moreover its market was not dominated by IBM or other manufacturers, because at the time of unbundling the product had not reached maturity, thus leaving ample opportunity for independent firms.

Applications were the largest sector, industry-specific the major part therein: “Measured either by number of packages or by market value, industry-specific applications constituted the largest sector of the software products industry” (p. 136). Martin Campbell-Kelly carefully warns the reader not to expect every detail of the multitude: “In the second part of this chapter, the software ‘product space’ is explored [...] Though far from exhaustive, the examples have been chosen to give a representative view of the world of software products” (p. 141). And as we come to the section of industry-specific application packages, the small letters say: “With several hundred firms by the end of the 1970s, the industry-specific software products industry was so fragmented that it is not possible to study a representative sample. This section, therefore, offers a ‘horizontal’ study of firms that offered software products for the banking sector and a ‘longitudinal’ study of a single firm (ASK Computer Systems) and a single application (Materials Resource Planning).” (p. 152) However hard it is to keep count of all such caveats, the reader must do so, or get lost.

Consolidators

Chapter 6 on ‘The Maturing of the Corporate Software Products Industry, 1980-1995’, indeed brings us Larry Ellison dominating the database market, but also a couple of surprises. The first, not so big surprise is that IBM, known to the world as computer manufacturer, is also the biggest software producer. The greater surprise to me was to see the role of consolidation presented as an utterly constructive one in the maturing software industry. This issue announced in the introductory pages of the chapter as Merger & Acquisition (p. 167). The section on Charles Wang and ‘Computer Associates, the Ultimate Consolidator’ (p. 178-185) offers a subtle analysis of what a consolidator, normally seen as a negative force buying himself into a dominating position in a market, may contribute and in fact did contribute to the maturing product: integration and rejuvenation. “To a degree, these acquisitions were opportunistic, but there was an underlying strategy of portfolio filling and product integration.” (p. 183) Campbell-Kelly illustrates

this strategy by Computer Associates'entry in the database market, to arrive at the general conclusion on CA's strategy: "After acquiring companies with low valuations but historically strong products and customer bases, it integrated the products into its own product line. [...] Computer Associates' has shown a unique ability to keep a portfolio of 500 products constantly rejuvenated and to give them the semblance of a product family." (p. 185) Together with the explanation of software as a product, as distinct from package, this explanation of the impact on the content of software of what is basically a market strategy, is one of my favourite parts of the book.
Salvation for the European soul in this strictly US-dominated narrative, is the final section of chapter 6 on the success of SAP ('BMW-quality' software) and Baan in software for ERP, Enterprise Resource Planning.

Debunking
The third part of the book, chapters 7-9, on the personal computer software industry shows Martin Campbell-Kelly at his best, not only presenting the reader with telling examples, but constantly debunking the industry's myths. Down goes the killer-application hypothesis, away with the impression that it is all Microsoft, dissolved the myth of Apple's own tradition –in fact we read that Microsoft gradually developed its Windows software on Apple computers before entering the IBM market with it. "Microsoft has truly dominated the personal computer software industry since the early 1980s. However, as this book has been at pains to point out, Microsoft still constitutes only about one-tenth of this extremely fragmented industry. Nonetheless, Microsoft's amazing success demands historical analysis. Was Bill Gates smart, lucky, or ruthless? A little of each, perhaps." (p. 264).
Chapter 9 on home- and recreational software gives due attention to a multibillion dollar sector, but adds less synthesis to the known narratives, than do the other chapters –some resulting from painstakingly reaping together disparate elements, some with a natural coherence in a fluent and passionate style.

Through the heterogeneity of the book there is a unifying element of style, exemplified by the above quotes: the recurring instances of almost unfiltered entrepreneurial recounts. The first one causes some amazement, the second a smile, the third a sad grin and from the fourth onwards just hichoughs. This unity of style is not a very attractive one. True, this is what the secondary sources will offer, but just as true, the historian does possess his filtering techniques. My objection is not just an irritation with the blurbs of heroism. As a historian I simply cannot believe that the economy of the late twentieth century is best described in a style deriving from the self-image of the industrial entrepreneur hundred years before. Even if one were to follow the author in calling the present economy microsoft-economy rather than post-industrial, one wants to keep some distance at this point.

2. Is it industry?

The style of presenting entrepreneurship like in the early industrial era may be inadequate for a more profound reason. It raises the pertinent question if it is *industry* we are looking at. With so many indications in the litterature and in this book itself, that labour relations and economic patterns diverge from those of the industrial era, are we right in describing the software sector as an industry.

Data

Given the scarcity of data, the author has done an admirable job, to bring together as much as he has done. Of course, any emerging sector of economy will be underrepresented in standard economic surveys. Economic analysts need time to adapt as well. What would be the adequate kind of data to collect on the software sector. The scarcity of data may well be just the surface of a more fundamental question.

Types of Economics

The problem seems to be rather persistent in this case. Indeed economists perceive a problem here as well. They have not agreed as to what kind of economy software is. It is not agriculture, nor heavy industry. Is it service economics, like banking, or insurance? Or is it more akin to the economics of care, like hospitals and welfare, or the economics of cultural goods? Certain goods in the software economy tend to behave like public goods, indeed there is a small but persistent movement among producers and users of software for the establishment of "commons", common grounds in virtual space like there used to be in traditonal villages.

At several points in his book Martin Campbell-Kelly does note the awkward behaviour of the economy of producing and selling software. He quotes the business press calling this economics of increasing returns "Microsoft economics" (p. 236), but speaks of industry allthesame. Here lies a fundamental question in the very domain of the book, which, I think, deserves to be addressed.

Fragmented

The presentation of software entrepreneurship is highly compartmentalized, in accordance with the author's insistence on the fragmented character of software business. He convincingly sketches so many markets, so many universes, that hardly communicate. Competition seems restricted to the proper fragment of the market. Cross-overs from one fragment to the other constitute the rare exception. Market fragments may die out.

Nonetheless, a unity of software industry, or software economy, is presupposed throughout the book. What constitutes this unity?

Thus on a conceptual level the book fails to address the implicitly assumed unity of the field. Also at an empirical level it is overdoing the fragmentation. If there were really no communication between all these fragments, and in each universe

the tendency towards quasi-monopoly yields, why are there not many more players in the software market, each with a universe, be it a niche, of their own? How come, in such fragmented market, a few very big players dominate accross these fragments? And how can the consolidators exert their integrating and rejuvenating force, if not across the borders of market fragments?
There is interaction between the universes, but under the given approach it is hardly allowed to appear.

Aspects of "Microsoft economics"
As long as software is delivered within the frame of consultancy, it is clearly part of a service economy. As a product however, with negligable production costs per item, it typically defies the economic law of deminishing returns. Instead the software economy is economics with *increasing returns*. Here is one aspect of "Microsoft economics"
Directly related to the increasing returns is a tendency towards *monopoly* for a certain product in its own market, or rather a tendency towards monoculture, a "winner-takes-all market" (p. 236). Here is another aspect of "Microsoft economics".
I would add –beyond what the litterature offers at this point– that by consequence software products dominating a market acquire a public character: everyone uses it and alternative choices lead to immediate social exclusion. Software products become *quasi-public goods*. A third aspect of "Microsoft economics".

Taken as industry, software business is deemed to appear as utterly fragmented. From the above, however, derive several good reasons to consider it as a unity. In fact such unity is a silent assumption of this book. The price to pay for reconciling the good reasons and the silent assumption will be not to see it as an industry.

Beyond
If we are allowed to break the frame of economic considerations and step outside the scope of this book, there is even more to be said about "industry". Further consideration of phenomena like the development of labour relations towards flexwork, like the (lack of) development of professionalism in software-related "work", like the quasi-public character of the produced goods, offers ample reason to reconsider the idea of the software sector being an industry. If one would, like Campbell-Kelly, rather accept "Microsoft economics" than "post-industrial society" as a denominator for the social order and economics of which software production is a pre-eminent part, then, first, this is somewhat in contrast with downplaying the importance of Microsoft to a mere 10%, and, second, it again presumes the unity of the sector.

3. Questions
Software appears as an economic good in this book. We come to view the evolution of software as a product, from services and contracts through packages

and products, to shrink-wrapped mass consumer goods. This is an invaluable gain for the history of computing. It is truly ground-breaking.
The book is setting an agenda in a natural way by provoking further research into the history
of the software sector. So many fragments, so many firms, so many entrepreneurs. It immediately calls for a complement studying the market rather from the consumer side.
Furthermore the book poses questions to the history of computing, the most fundamental of which is for other approaches to come up historical accounts of software meeting this economic historical view.
Is there an approach to software in the history of technology to meet these results? Are there history of science -, history of culture-, history of consumption approaches yielding an account of software confirming or denying this narrative?
Could we see software emerge in the 1950s as a technology that would fit the early economic appearance of it, as presented here by Martin Campbell-Kelly? If so, what was its content? What were these assembly systems, operating systems, and compilers presenting the software delivered under contract? What part of software was being re-used, what were programming libraries of the 1960s in relation to those of the 1950s –to mention a less convincing detail of the book–? What were these 10,000 lines of code presenting an operating system for the IBM 650; the 100,000 lines for the IBM 1401; the 1,000,000 lines of codes OS 360 for the IBM System/360, and 10,000,000 for the 370?
Can we see a technical evolution of software to better understand the transition from package to product in the late 1960s? Was software indeed improving, or were consumers simply more effectively disciplined to accept the products?
And most urgently, of course, can cultural historians tell us if indeed it is an industry?
Recent work by Yates, Haigh, Mahoney and the essays in Hashagen's book *Software Issues* –Campbell-Kelly wrote one of these essays– all contribute fragments to the answers of such questions. Now that Martin Campbell-Kelly has taken such a leap ahead, let us take up the challenge.

Reply to Adrienne van den Bogaard, Frank Veraart and Gerard Alberts

Martin Campbell-Kelly

I should perhaps start with a few words about how I came to write *From Airline Reservations to Sonic the Hedgehog*. In 1994 when I was writing *Computer: A History of the Information Machine* with Bill Aspray, we needed to say something about the software industry. To our dismay almost nothing had been written. There were two major volumes and several academic papers on programming languages, but on the much more important topic of the software industry there was very little indeed, and certainly nothing written by a professional historian. To fill the lacuna, and to force myself into doing some basic research on the topic, I submitted a paper "Development and Structure of the International Software Industry" to the Business History Conference in spring 1995. The paper was subsequently published in *Business and Economic History*.[1] In this paper I proposed a three-sector model for the industry, which several individuals subsequently told me they found a unifying and helpful way to think about the industry. The three-sector model was an original idea (not a very deep one, I agree) but it had not been suggested before. Like all historical simplifications (such as the Renaissance or the Industrial Revolution) it was a fiction, but there is an unstated agreement among historians that we accept such constructions when they enable us to understand what would otherwise be incomprehensible chaos. In effect *From Airline Reservations to Sonic the Hedgehog* is a fuller and more richly detailed extension of that paper.

In their review, Adrienne van den Bogaard and Frank Veraart succinctly explained the three-sector model, as consisting of the triad of software contractors, producers of corporate software products, and makers of mass-market software; and that "[w]ithin one sector companies compete against each other; among the sectors there is hardly any competition—they are more or less mutually exclusive because of different markets and capabilities." This is the central thesis of the book, and most of the book is really scaffolding to support this proposition. I never had any higher ambition for the book than that people would buy into this proposition and use it as a basis for future historical studies in the software universe. Many of the book's failings arise from the difficulty of finding sufficient scaffolding and trying to tell a very big story in just 300 pages.

I've been gratified by the many divergent reviews my book has received. Authors love to get attention and—perhaps speaking here just for myself—a bad review is better than no review. My worst review by far came from Professor Harold Perkin the distinguished social historian who concluded his review with the remarks:

[1] M. Campbell-Kelly, "Development and Structure of the International Software Industry, 1950-1990", *Business and Economic History* **24**, 2, pp. 73-110.

"Campbell-Kelly is a master of the technical detail and the alphabet soup of acronyms but, like most specialists in an arcane activity, he has tunnel vision and provides little social and economic context. He does "internalist" history, rather like old-fashioned art history or history of science, full of innovators and heroes driven by creative opportunity. The impact of the computer industry on society, on the way people live and communicate, is largely left to the reader's imagination. ... This technically expert book is rather like old railway history written by railway buffs who know the number of wheels and the horsepower, the name of the engineers and companies, but take for granted how they changed the world."

I think Perkin overstates my internalism, although what he says has substance. But the book he wants is not the one I set out to write. It is as if Perkin, having read a history of the supermarket industry, is disappointed not to find an explanation of how Jello and radio advertising combined to redefine the dessert experience of American children in the 1940s. My book is a supply-side history of traded software. It is not a cultural history of software, nor a scientific history. I would love to read books on these topics, but they are subjects that my book only touches upon, and often only in passing to add colour to a dull narrative.

The thirty or so reviews of my book are generally positive, but not ecstatic. I have come to accept that I have failed to please all of the readers, all of the time. Some authors do come close to that ideal, but I am nowhere near. Perhaps these other authors have more tractable material; more likely they are better authors.

It is very rare that an author gets the chance to reply to reviewers—other than by writing a peeved letter to the editor of the offending publication—so I really appreciate the forum my friends in the Netherlands have given me to respond to their reviews. Your comments are especially valuable to me because they come from individuals who are immersed in the history of computing and therefore have a much better grasp of the challenges than some of my other reviewers. Naturally I don't agree with all the observations, but I am inclined to the view that I have generally failed to explain my purpose clearly, rather than that you have been perversely myopic.

This long preamble has been a way of reconciling two opposing facts. First, that your criticisms have substance. Second, that I stand by the broad structure of my book and would not do it much differently if I were starting again. I wish I could have written a better and more readable book, to be sure, but that is a different issue.

Gerard Alberts asks the question "Is it industry?" In a trivial sense it is—my book describes a collection of firms that sell software. However, Gerard is more concerned about the coherence of the framework I have presented and the availability of data to fill out that framework. This is a valid criticism—it is as if I had written a book called *The Retail Industry* that tried to incorporate in a single framework department stores, supermarkets, home-improvement emporia, right through to mom-and-pop corner stores and hot-dog sellers; and moreover, that my data on hot-dog sellers was decidedly thin. The only unifying theme would be that these enterprises sold goods to the general public. My book is somewhat like this—I have tried to explain the universe of traded software, but the largest and smallest enterprises in this universe are as different as a department store and a hot-dog seller. My aim in trying to be so inclusive was the better to set firms such as IBM and Microsoft—and the ten-person software firm across the street—in a better context. I think the world needs the occasional over-broad book, whether it's about the retail industry, world religion, or software. I don't expect anyone to repeat the exercise any time soon. I'm sure the next person to write a longitudinal history of the software industry would be wise to focus on a single sector or software genre.

Gerard is also concerned about my repetitious use of vignettes of individual firms and entrepreneurs to carry the narrative:

"[T]he recurring instances of almost unfiltered entrepreneurial accounts. The first one causes some amazement, the second a smile, the third a sad grin and from the fourth onwards just hiccoughs."

Gerard is correct when he supposes that these sketches were a cover for lack of statistical data, but it was also a conscious decision—a way of telling a story in a manner that would reach out to a readership beyond the academy. I was following a writing style that Bill Aspray and I developed for *Computer*. For this book we benefited from much practical advice given to us by our editor, a seasoned shaper of semi-popular science, technology, and business books. She encouraged us to tell our story through a series of vignettes—of individuals, inventions, technologies, and firms. We should not attempt to tell the reader everything, but instead by the judicious choice of vignettes enable him or her to integrate the examples into a coherent narrative. I tried to avoid excessive hero worship or lurid tales of boardroom coups. To take one example, Gerard cites the use of the tired adjective "charismatic" for Fletcher Jones, the founder of CSC. Actually, I was toning down some exceptionally gushing sources, such as this one from the *New York Times*:

"Mr. Jones is a Gary Cooper type—Texas-tall, lean and soft spoken. ... Computer Sciences' founder, president and chairman is also the prototype of the "jet age" executive. He is young, 37 years old; articulate, always on the go and apparently unhampered by corporate red tape."[2]

On the whole readers seemed to want more, not less, of this type of colour and they were critical of the statistical dryness of the book and its intimidating tables. Thus, Steve Lohr wrote in the *New York Times*:

"Campbell-Kelly's book is not for everyone. He is an instructor in computer science at Warwick University in England and is a professional historian. His is not a book of insider gossip or of recreated scenes of clashing egos and executive tirades—the stuff of so many business books. Instead, it is a product of his reading and distilling of books,

[2] William D. Smith, "Texan Guides Software Unit to Big Board," *New York Times* (December 1, 1968): 3.

professional journals, magazine and newspaper articles and historical archives over the last four decades. ... He is fastidious with footnotes."

Some reviewers found me uncritical to the point of obsequiousness. After all I was neutral on Bill Gates and only mildly disrespectful of Larry Ellison. Other reviewers rapped my knuckles for omitting their favourite episode in the history of software.

So I guess the book hit the middle ground—pleasing neither the academy nor the general reader very greatly. However, the book has reached both readerships and even in semi-popular writing I already see evidence of the adoption of the three-sector model and a rejection of the myth that Microsoft was the progenitor of the software industry.[3]

In their review, Adrienne and Frank comment on my unbalanced account of programming languages. Specifically, my dragging out the tale of FORTRAN and slighting JOVIAL. This was one of many conscious judgments that were to do with proportionality. At its peak, FORTRAN accounted for perhaps 80 or 90 percent of scientific programming around the world and it is still very influential. So it, along with COBOL, deservedly got a two-page treatment. JOVIAL, which got a one paragraph discussion, was very important in U.S. military contexts, but never diffused to the civilian sector. ALGOL, on the other hand, had no industrial significance in the United States (apart from a short love-affair with Burroughs Computers in the 1960s) and relatively little in Europe, and so did not feature in the book at all. This was not a scientific process of course—there was no master list of programming languages to ensure that each minority language got a proportionate mention.

In fact the principle of proportionality governed the whole of the book. I devoted about ten percent of the space to Microsoft because that is Microsoft's current market share. Indeed, in the 50 year history of the software industry Microsoft is something of a late-comer, so I was actually rather generous—but I felt that anything less than 10 percent would be too big a pill for the PC-centric reader to swallow. The principle of proportionality also explains why major firms such as IBM, Computer Associates, SAP, Oracle, and Microsoft get very much more attention than small companies, which are generally dealt with in the aggregate. I have behaved rather like a military historian who writes about the generals and ignores the poor bloody infantry. You are right to criticise me for my cavalier disregarded of minority subjects. That said, I personally know of no other way to write a coherent narrative than to exercise such judgements. I think the place for extended discussions of JOVIAL or ALGOL is in the periodical literature, where there is less of a need to present a big picture.

Not surprisingly, my Dutch reviewers discern a social constructive perspective in my portrayal of the Darwinian selection of winners and losers in the software

[3] See, for example: Detlev J., Hoch, Cyriac R. Roeding, Gert Purkert, and Sandro K. Lindner, *Secrets of Software Success: Management Insights from 100 Software Firms Around the World* (Cambridge, MA: Harvard Business School Press, 1999); Wilson, Mike, *The Difference between God and Larry Ellison: Inside Oracle Corporation* (New York: Morrow, 1996).

eco-system. However, you point out that I give only "ad hoc arguments" and never satisfactorily explain why one firm succeeds while another fails. Mea culpa. The fact is I really don't have a wholly convincing explanation that goes beyond the particularities of accidental capabilities, individual entrepreneurial genius, and chance events. However, there is the germ of a more promising explanation buried on page 265 where I wrote:

> "Popular writing on the PC industry often appeals to the idea that luck—being in the right place at the right time—was a major factor. This is a theme suggested by the title of Robert Cringely's excellent and entertaining book *Accidental Empires*, for example.[4] Luck did indeed play a big part. The understanding of increasing-returns economics was luck—the practitioners who invented or stumbled across this way of doing business prospered, whereas others who held to some different belief (say, that high-quality software or low-cost packages would lead to market success) bet on the wrong theory, and were weeded out in the Darwinian evolution of the industry."

In short, one had to have the right theory, some luck, and do almost everything right. I think that successful entrepreneurs have an idiot-savant quality that most people do not possess: they have a genius for seeing what others do not see, and what they see corresponds to reality. In one of my interviews with Arthur Humphreys—undoubtedly the best CEO that the British computer firm ICL ever had—he said that the essential quality of a chief executive was an ability to make great decisions with very uncertain information.

This is, of course, a rather naïve explanation, and it offers little hope to people without entrepreneurial second sight. It is certainly not a view that would be subscribed to by writers such as Tom Peters, whose *In Search of Excellence* offered practical lessons for would-be entrepreneurs.[5] But one should bear in mind that within a few years of publication, several of Peters' high-tech case studies—Amdahl, Data General, DEC, and IBM—had hit the rocks.

To address Adrienne and Frank's last comment, my discussion of how firms compete with Microsoft was not intended as a policy recommendation, it was simply an observation. It was a reactive response to the fact that some conscious or unconscious strategies work for some firms. Despite the advice in *Secrets of Software Success* I do not think there is a sure-fire recipe for success. In the Darwinian soup of the software industry, no one knows for certain the best way to avoid being stepped on by roaming dinosaurs such as IBM, Computer Associates, or Microsoft.

[4] Robert X. Cringely, *Accidental Empires: How the Boys of Silicon Valley Made Their Millions, Battle Foreign Competition, and Still Can't Get a Date* (Cambridge, MA: Addison-Wesley, 1992).
[5] Thomas J. Peters and Robert H. Waterman Jr., *In Search of Excellence: Lessons from America's Best-Run Companies* (New York: Harper Collins, 1982).

References

Campbell-Kelly, Martin, *From Airline Reservation to Sonic the Hedgehog; A History of the Software Industry*, Cambridge, MA: MIT Press, 2003

Campbell-Kelly, M., ‘Development and Structure of the International Software Industry, 1950-1990’, *Business and Economic History* **24**, 2, pp. 73-110.

Campbell-Kelly, Martin and William Aspray, *Computer: A History of the Information Machine*, New York: Basic Books, 1996.

Robert X. Cringely, *Accidental Empires: How the Boys of Silicon Valley Made Their Millions, Battle Foreign Competition, and Still Can’t Get a Date,* Cambridge, MA: Addison-Wesley, 1992.

Hashagen, U., R. Keil-Slawik, and A. Norberg (eds.), *History of Computing: Software Issues*, Berlin/Heidelberg: Springer-Verlag, 2002.

Hoch, Detlev J., Cyriac R. Roeding, Gert Purkert, and Sandro K. Lindner, *Secrets of Software Success: Management Insights from 100 Software Firms Around the World,* Cambridge, MA: Harvard Business School Press, 1999.

Haigh, Thomas, ‘Software in the 1960s as Concept, Service, and Product’, Thomas Haigh. *IEEE Annals of the History of Computing*, January-March **2002**, 5-13.

Peters, Thomas J., and Robert H. Waterman Jr., *In Search of Excellence: Lessons from America’s Best-Run Companies*, New York: Harper Collins, 1982.

Smith, William D. ‘Texan Guides Software Unit to Big Board’, *New York Times* (Dec. 1, 1968): 3.

Wilson, Mike, *The Difference between God and Larry Ellison: Inside Oracle Corporation,* New York: Morrow, 1996.