Frank Land

Book review: the technical and social history of software engineering

Article (Accepted version) (Refereed)

Original citation:
Land, Frank (2015) *Book review: the technical and social history of software engineering.* Communications of the Association for Information Systems, 36 . pp. 577-581. ISSN 1529-3181

The Technical and Social History of Software Engineering
Capers Jones, Addison Wesley, 2014

Frank Land, London School of Economics

Review Article

A recent book promised from its title and authorship to fill a long-standing need for a text which would gave readers of CAIS an understanding of the history of software engineering. This article sets out to examine the extent to which the author succeeds in meeting that promise. As such it is intended for the IS History section of CAIS.

The author Capers Jones has a long record of making significant contributions to software metrics and the economics of software, including the publication of 15 books related to these topics. Hence his new book on the technical and social history of software engineering was eagerly received by this reviewer.

The book is long - 452 pages – but the subject is large and indeed as the author notes, if his history had been fully comprehensive, it would have required at least 1000 pages.

In this book the author has set out to chronicle the innovations, the entrepreneurs, the organizations, private and public, associated with the evolution of Information Technology and its software. His starting point is the Stone Age and his story finishes in 2013 though he ventures some predictions up to the year 2019. Although the focus is primarily on the American experience the author does make an attempt to bring an international flavour to his account.

A prelude takes the long view, providing lists of innovations in mathematics, recording and calculating from the beginning in the Stone Age, up to 1930. Subsequent chapters take each decade from 1930 to 2019. The final chapters, chapters 11 and 12, review the problems and failings of software engineering and report on the threats posed by cybercrime and cyberwarfare.

The story told is often interesting and illuminating, but overall the book is a disappointment. I would expect a history of software engineering to provide an account of ideas behind the way software engineering developed including accounts of some of the debates such as that triggered by Edsger Dijkstra's letter in the Communications of the ACM on the Go To statement.[1] Vannever Bush's 'invention' of the Memex[2] published in 1945 is not mentioned though it was an important milestone and surely ranks close to Turing's paper on computability and in its anticipation of hypertext and thus the Worldwide Web Many more examples could be cited. The book is not the history of software engineering, technical or social, that a student of the subject might expect.

[1] Edsger Dijkstra *Communications of the ACM*, Vol. 11, No. 3, March 1968, pp. 147-148.
[2]http://en.wikipedia.org/wiki/Memex

Again no mention is made of the landmark program by Daniel Teichrow[3], first at Case Western, later at the University of Michigan in the 1960s, to devise a system which would automate the production of computer code directly from the requirements specification .The project named Information System Design and Optimization System (ISDOS) developed a special Program Specification Language (PSA) and an associated Problem Statement Analyser (PSL) followed by a code optimization process (SODA). The system was successfully deployed in industry but also had important impact in education and in triggering much fruitful research.

Another very important thread that is largely missing from the text –is the critical role played by Government and politics in the evolution of IT and its enabling software. True, developments in defense computing are discussed throughout the book, but the larger role of Governments is neglected. A good example is the Japanese Government sponsored project known as the 5th Generation in 1982[4]. This $850 million project was intended to lift computing into the new realms of knowledge based AI systems built on improvements in both electronics and software engineering. The success of Japan at that time, with its increasingly dominant role in automobiles and electronics, led Governments and industry world-wide to fear the future dominance of Japan in computing. The result was that the US (under DARPA), the UK (Alvey)[5], France (Plan Calcul[6]), and the European Union (ESPRIT[7]), each sponsored major multi-million dollar research projects which brought academia, industry and Government into collaborative research schemes. In the UK, for example, the Alvey programme, launched in 1983 with funding of ₤350 million over 4 years for industry/academia projects included:

- VLSI (Very Large Scale Integration) technology for microelectronics
- IKBS (Intelligent Knowledge Based Systems) or Artificial Intelligence
- Software Engineering including the design and construction of Computer Aided Software Engineering (CASE) tools
- Man-Machine Interface (included Natural Language Processing)
- Systems Architecture (for parallel processing)

Other Government initiatives in various countries with far reaching consequences were concerned with the structuring and restructuring of their computer industries. This included the attempt in the 1970s to bring together the major players in the Dutch, German, French and UK computer industries as one company, UNIDATA[8]. The attempt ultimately failed, partly because of the reluctance of the UK's major player ICL to join the consortium.

In 1966 Soviet Union authorities, noting the success of the IBM 360 series world wide, took the momentous, but heavily criticized, step to abandon the Soviet's indigenous

[3] http://www.pslpsa.com/index.php/79-isdos-category/90-dt-article
[4] http://en.wikipedia.org/wiki/Fifth_generation_computer
[5] http://hansard.millbanksystems.com/written_answers/1988/jan/21/alvey-project
[6] http://en.wikipedia.org/wiki/Plan_calcul
[7] https://aclweb.org/anthology/H/H91/H91-1007.pdf
[8] http://www.feb-patrimoine.com/histoire/english/chronoa8.htm

computer designs and replace them with IBM clones[9]. The Russians managed to clone the hardware quite well but had more problems with the software engineering, failing to properly re-engineer IBM systems software. At that time an embargo (CoCom[10]) initiated by the US Government and its allies as part of its cold-war strategy prevented American computer technology from being exported to the Soviets and its allies.

Although most of the initiatives, including the Japanese 5th generation project, failed to achieve their major targets, they left a lasting legacy in terms of collaboration across borders and between Universities, research establishments and industry.

One further Government sponsored initiative which should be noted in a history of software engineering was the launching in France and the UK of the Minitel[11], Ceefax[12] and Prestel[13] teletext service which provided subscribers with an information service direct to their television screen. This anticipated much of the web-based services we now rely on. The French system Minitel had considerable success with millions of subscribers, in part due to the subsidies provided for subscribers and lasted well into the 21st century. The uptake of Prestel services was more limited and faltered earlier and the service was soon abandoned but Ceefax ran well into the 21st century.

In his final chapter (chapter 12) Capers Jones discusses the rise and rise of Cybercrime and more briefly notes the future of cyberwarfare. Unfortunately the book was written just too early to catch the Edward Snowden revelations of the joint mass surveillance of its citizens by Government agencies in the US and the UK. The advance of computing, and communication capabilities all enabled by software engineering have provided the means by which suitably equipped agents can capture details of all calls made on mobile phones, email messages, data stored, searches made via browsers like Google, contents of Facebook and Twitter messages, and can even decrypt coded information. Snowden's revelations indicate that Government agencies capture significant proportion of all internet traffic. The purpose of this mass surveillance is security against terrorist attack and criminal intent.

Any further editions of the book will surely include a discussion of the privacy versus security aspects of the Snowden revelations, including the security of the surveillance agencies themselves in the light of Snowden's discoveries[14].

There are a number of other omissions which a history might be expected to cover. These include the establishment of the International Federation of Information Processing (IFIP) by Isaac Auerbach[15]. IFIP has a number of Technical Committees each made up of international working groups covering a range of topics including computer languages, software engineering, different types of systems, and all aspects of computer education,

[9] http://en.wikipedia.org/wiki/ES_EVM
[10] http://en.wikipedia.org/wiki/CoCom
[11] http://en.wikipedia.org/wiki/Minitel
[12] http://en.wikipedia.org/wiki/Ceefax
[13] http://en.wikipedia.org/wiki/Prestel
[14] http://en.wikipedia.org/wiki/Edward_Snowden
[15] http://en.wikipedia.org/wiki/International_Federation_for_Information_Processing

as well as computing history – working group 9.7. There is no question that some of the working groups have made major contributions to their fields of study, including the field of software engineering.

Coverage of the role played by international standards organizations[16] such as ISO and the influence they have wielded in the evolution of software practices – both positive and negative - are treated very lightly and not in a systematic manner. Another example is the much discussed Productivity Paradox which emerged as a concept in the 1990s, though the economic value of investing in business computer systems had been listed as a top ranking concern of business leaders in survey after survey since the 1960s[17].

Too often the treatment of topics is poorly balanced. Thus the very important topic of Enterprise Resource Planning (ERP)[18] which transformed the way many companies arranged their information processing and the role of their in-house software engineers is only mentioned in passing – receiving far less coverage than the Norden bombsight but whose actual impact on software engineering is at best slight. There are many examples of poor judgments on which topics to require more or less emphatic treatment. The problem is exacerbated by the author's use of long lists with the list of Beneficial Tools and Applications (page 23) covering more than 9 pages. It is as if the author has forgotten all his own stress about the importance of function points and reverted to lines of code.

The fascinating Antikythera mechanism, possibly invented by Archimedes, which had the capability of predicting solar and lunar eclipses and whose identification resulted in a complete reappraisal of the mathematical and engineering expertise of the ancient Greeks is listed in table 1.1 without any explanation. Inventions such as double entry book keeping are not mentioned or commented on and neither is the re-invention of movable type by Gutenberg around 1450[19] – perhaps as important in the late Middle Ages as the invention of electronic computers in the 20th century.

Missing too from the early years is any discussion of the sophistication reached by automata in the 16th, 17th and 18th century including the importance of arranging the complicated movements of the figures by means of a fixed program determined by the shaping of a cam and later the pattern of holes in a card driving musical instruments and industrial machinery as in the Jacquard Loom. No mention is made of the 1951 invention of micro-programming by Maurice Wilkes[20], though the technique became central to the architecture of the IBM 360 range providing an essential tool for permitting compatibility across the range.

To avoid repetition see Robert Glass's review of the book in the Spring 2014 edition of the **Software Practitioner** providing a further critique of the book including naming

[16] http://www.stanhopecentre.org/cotswolds/IT-Standardisation_Jakobs.pdf
[17] http://ccs.mit.edu/papers/CCSWP130/ccswp130.html
[18] http://en.wikipedia.org/wiki/Enterprise_resource_planning
[19] http://en.wikipedia.org/wiki/Johannes_Gutenberg
[20] http://en.wikipedia.org/wiki/Microcode

more omissions. Robert Glass writes: "*My major thoughts on what was missing from Capers' book are:*

1. There is no mention of the early business computers and their programming. They were decimal, variable word length, and organizationally separate in industry from their scientific brethren (both in the computer facility and the programmers' locations). They were also very separate product lines at such companies as IBM. When the 360 did away with that division, the social implications were powerful and troubling.
2. Stored-program computers. The earliest IBM machines, after board wired calculators, could not store the program in memory. Instead, the Card-Programmed Calculator (CPC) executed each instruction as it read the card containing it via the card reader. Looping and branching were hugely complicated programming tasks in this mode! The notion of stored programs was a major change in how programming and computing was done."

Names of important contributors are often listed with only the soubriquet 'famous' before the name and without any explanation what the fame is based on. The reader who has not come across that name but is interested has to search the internet to find out what the named person is famous for.

More important each chapter has a table - see for example table 8.3 on page243 – giving a breakdown of growth of Software Applications in the U.S. in the decade covered by the chapter. No information is provided on how such a table was constructed or the sources of the breakdown. The individual items are often ambiguous. What, for example, are Information technology applications? What counts as an application? How accurate is the breakdown and the numbers?

The author does not make clear who the book is written for. Reading it does not answer the question. For the student of software engineering the book provides little to help for the student to understand the ideas or practices which underlie software engineering. A scholar would be concerned at the lack of citations to sources and the many omissions. The 'layman' reader might be irritated by the long lists and the lack of explanations.

And yet the book has many virtues. It provides a readable account of the way the IT industry has grown from very small beginnings both in range of application and in the depth of its penetration into every aspect of our lives world wide. The later chapters on the threats posed and the many failures are excellent. I liked Capers Jones's reminiscences of his early days as a programmer at IBM in part because they mirrored my own experience with LEO computers a decade earlier.

Perhaps it would require at least two volumes to provide the kind of history which of Software Engineering I would like to see. Is this a challenge Capers Jones is willing to meet?