A Course on Revolutionary Computing Ideas of the 20th Century

G. K. Gupta
Faculty of Information Technology
Monash University, Building 75
Clayton, Australia 3800
(email: gopal@infotech.monash.edu.au)

BIOGRAPHY

Professor Gopal Gupta completed BE(Hons) at the University of Roorkee (now, IIT Roorkee), India, a Master's from the University of Waterloo and PhD from Monash University. He is currently Professor of Computer Science, Monash University. Previously he was Professor and Head of Computer Science, James Cook University and Dean of Information Technology, Bond University. Major areas of interest: database systems, biometrics and computer science education.

ABSTRACT

It appears that many Computer Science graduates do not get sufficiently acquainted with some of the greatest ideas of the discipline during their undergraduate studies. I present a subjective list of great computing ideas of the 20th century and recommend that a course during the final year focusing on these ideas, or a similar customised list of ideas that better meets the needs of a program, be included in all Computer Science curricula. A course like this could serve a least three primary aims, namely, to get the students acquainted with some computer science history, to raise students' awareness of the great computing achievements of the 20th century and finally to present to students an overall unified view of the discipline before they graduate.

Categories and Subject Descriptors
K.3.2 Computer and Information Science Education

Keywords
Great ideas in Computer Science, Computer Science curriculum

1. Introduction

Over many years of teaching Computer Science (CS), I have often been surprised to find that many CS graduating students lack understanding of some of the greatest CS ideas. To test this perception, I recently carried out an informal survey of a final year class of CS and Software Engineering (SE) students. One of the two questions asked was:

> Write the names of no more than five important computer scientists who lived during 1940-1990 (and may be alive today) that you have heard about.

The class had 60 students all of whom completed the survey. The following results were obtained. Names suggested by four students or less are not included.

Name	Number of Students
Turing	38
Dijkstra	27
von Neumann	13
Gates	12
Backus	7
Church	7
Stroustrup	6
Kolmogorov	6
Warshall	5
Mealy/Moore	5

The only computer scientists that more than 10% of the students could recall were Turing, Dijkstra, von Neumann, Bill Gates, Backus and Church. Only Turing and Dijkstra were identified by about 50% or more. A number of names given (for example, Kernighan, Deitel, Stallings) were authors of textbooks that were used in the CS and SE undergraduate courses. Although this was not a scientifically conducted survey, I suspect that a more scientific survey of graduating students at our institution could well lead to results similar to those obtained and it would not be surprising if these were typical results for graduating CS students in Australian universities.

The second question in the informal survey related to important ideas of Computer Science. The question was:

> Write no more than five important computing ideas that you have come across in your studies so far.

There were more than 100 ideas suggested and in hindsight the question could perhaps have been better designed. Each of the ideas in the Table below was suggested by more than four students. No idea was suggested by more than about a quarter of the students. No one suggested, for example, stored program computer, Shannon's information theory or relational data model as important ideas. Many of the "ideas" suggested are clearly CS topics rather than specific ideas.

Name	Number of Students
Object Oriented	15
Turing machine	11
Artificial intelligence	9
Algorithms	7
Data structures	7
GUI	6
Sorting (smart algorithms)	6
Networks	5
Operating systems	5

For the CS community, there is a list of Turing Award winners and their talks and citations that provide a good basis for developing a list of major contributions to CS although the awards started only in 1966. Also, there are at least four books, Biermann [1], Dewdney [4], Laplante [8] and Pylyshyn [11], that discuss important CS ideas. These books are useful resources but none of them appear suitable for an advanced course. For example, Dewdney [4] briefly discusses 66 ideas. Such a large number of ideas would be difficult discuss them in depth since not even one lecture could be devoted to each of them.

A number of proposals have been made in the last 10 years about the importance of history of computer science in the CS curriculum. For example, Lee [9, 10] has forcefully argued for history to be included in the CS curriculum. Computing Curriculum 1991 and Computing Curriculum 2005 have included material on history in a number of courses, for example, artificial intelligence, operating systems, programming languages, and social, ethical and professional issues. Although, many important ideas are covered in the various courses, for some reason students still appear to lack a good overall view of the discipline and its intellectual history.

An IFIP TC3 and TC9 task force has presented an outline of a computing history course (Impagliazzo *et al* [6]). The proposal suggests three possible different approaches to teaching computing history. First, CS history may be integrated throughout the computer curriculum, second it may be a separate course designed for computing majors and finally it may be a course that is available to all non-CS majors. A list of topics for a CS history course is presented. One suggested approach for presenting the material is to divide computing developments into seven clusters, for example, mechanical computing devices before 1930, the mainframe era, evolution of architectures. CS history course syllabus samples from some universities are included.

Eisenberg [5] has proposed a course on "classic" readings of Computer Science that he suggests could be taken by senior CS students. A list of about 30 papers that are at least 20 years old with a bias towards artificial intelligence and human-computer interaction is presented. Eisenberg notes that the proposed course should not be considered a history of computing course.

Cortina and McKenna [3] discuss a history of computing course designed for CS majors as well as non-majors that involves lectures by a number of faculty members and video presentations every other week. The course included historical, political and social events and it examined lives of some of the most

influential CS inventors and thinkers. In addition, social, legal and ethical issues were included.

The course proposed in this paper is different from computing history courses outlined above because the focus of the proposed course is intellectual history of computing of the 20th century rather than history of computing which usually focuses more on important historical events.

A list of ideas for the proposed course was prepared after considerable consultation with half a dozen senior colleagues. The list, which grew to more than 100 long, was reduced to 25 ideas. I call these ideas "revolutionary ideas" of computing. The rationale for this is presented in the next section. I believe these ideas can be the basis for an excellent course which could be presented in the final year of an undergraduate CS program. It may also be a useful introductory course for graduate students.

The course proposed could have a number of objectives. It clearly is designed to provide the students an overview of some of the greatest ideas in the discipline. It can also help motivate teaching of a number of CS topics and enliven some of the CS courses by also covering some social aspects of the CS innovators. The primary aims of the course therefore should be: to get the students acquainted with the history of the greatest computing ideas of the 20th century and to motivate them in a number of CS topics and for graduate studies.

The paper is organised as follows. Section 2 presents the rationale for selection of ideas in the list and presents a chronological list of 25 ideas. Section 3 sketches one approach to presenting the course. Section 4 deals with two rival approaches. Section 5 presents conclusions.

2. Rationale and the list of revolutionary ideas

The Oxford English Dictionary defines a revolution as "involving or causing dramatic and far-reaching change or innovation. To change radically or fundamentally". Therefore to be considered a revolutionary idea, the idea must have resulted in:

- Dramatic and fundamental change
- Far-reaching impact

A revolutionary idea therefore has to have a very broad impact on the discipline and should be critically important where it applies. According to Kuhn [7], scientific revolutions are those non-cumulative developmental episodes in which an older paradigm is replaced in whole or in part by an incompatible new one. Why should a change of paradigm be called a revolution? Again, according to Kuhn, scientific revolutions are inaugurated by a growing sense that an existing paradigm has ceased to function adequately in the exploration of an aspect of the discipline in which that paradigm itself had previously led the way. In both political and scientific developments the sense of malfunction that can lead to crisis is prerequisite to revolution. The computing ideas in the table that follows were chosen keeping Kuhn's view of scientific revolutions in mind.

Innovations that are only used by a very small area of computing even if that area is important are not included in the list. Ideas from early history of computing, for example ideas of Boole and Babbage, are also not included.

	Year	Revolutionary Idea	Person Responsible
1.	1931	Incompleteness	Gödel
2.	1936	Turing Machines	Turing
3.	1945	Hypertext	Vannevar Bush
4.	1945	Stored Program Computer	von Neumann
5.	1947	Transistor and Integrated Circuits	Bardeen, Brattain, Shockley and Kilby
6.	1948	Information Theory, Entropy	Shannon
7.	1950	Computing and Intelligence	Turing
8.	1953	Hashing	Luhn
9.	1957	Fortran and its Compiler	Backus *et al*
10.	1960	Lisp and Functional Programming	McCarthy
11.	1960	Algol 60 definition and BNF	Naur *et al*
12.	1961?	Packet Switching	Baran
13.	1963	Image as Data, Image Manipulation	Sutherland
14.	1964	IBM System/360	Amdahl, Brooks and Blaauw
15.	1965	Computational Complexity	Hartmanis and Stearns
16.	1966	Simula and OO Programming	Dahl and Nygaard
17.	1967	Information Privacy	Westin
18.	1970	Relational Data Model	Codd
19.	1971	NP-Completeness	Cook/Levin
20.	1971	Unix operating system	Ritchie and Thompson
21.	1973	PC architecture and GUI	Thacker *et al*
22.	1978	RSA Algorithm, Digital Signatures	Rivest, Shamir and Adleman
23.	1979-80	PC Software – Spreadsheet, WP and Email	Several
24.	1989	World Wide Web	Berners-Lee
25.	1990	Search Engine	Emtage

This is a subjective personal list of 25 revolutionary ideas. Ten of them have been recognised by Turing Awards. A number of contributions were made well before the Turing Award started in 1966. When this list has been presented in seminars in Australia and overseas the audiences have usually accepted most of the ideas in the list. Discussion has often focussed on two or three ideas, whether they really belong to the list or if they could be replaced by some other, more important, ideas. Therefore I expect readers to disagree with some of the selections but hope they will agree with most of them. Lecturers teaching the course could obviously use this list to build their own list of favourite ideas, which could well include more than 25 ideas. The lecturers should be comfortable with the course curriculum.

3. One approach to presenting the course

The 25 revolutionary ideas and presentation of a course based on them are now discussed. A list of original papers where the ideas were published is presented in the Appendix. These original papers as well as other supporting materials may be used in the course. It should be noted that discussion of these ideas should be combined wherever possible with profiles of people involved. Discussing the lives of people like Gödel, Turing, von Neumann, Shockley, Vannevar Bush and Shannon adds a human touch which makes the course interesting, even exciting.

Furthermore, when considering the most important ideas from the 1950s and before, the lecturer should be able to relate these ideas to how computing developed in the early days from abstract ideas to building of the first computers followed by development of the first programming languages.

We now present the first six ideas from the list above and suggestions on how each of them can be presented including some details about their inventors. Much more information about them and others is available in computing history books and on the web (for example, wikipedia.org is a valuable source of information). For some important figures, for example Kurt Gödel, Alan Turing and John von Neumann, biographies have been published that are quite useful.

INCOMPLETENESS

This idea is introduced by noting that in the 1920s David Hilbert posed a number of questions about the core of mathematics. He wanted to find a general algorithmic procedure for answering all mathematical inquiries, or at least proving that such a procedure existed. In 1928, 22-year old Kurt Gödel attended a lecture by Hilbert in Bologna on completeness and consistency of mathematical systems. That must have sparked his interest in the topic as in 1931, when he was only 25, he proved the incompleteness theorems. A simple way to express the main result is:

> In any formal system adequate for number theory there exists an undecidable formula – that is, a formula that is not provable and whose negation is not provable.

Gödel completed a PhD from Vienna and then became a faculty member there in 1930. In 1934, Gödel lectured at Princeton and again in 1938-39 when the Second World War started he left Austria permanently for Princeton. Gödel was a strange genius. Amongst his delusions was the belief that unknown villains were trying to kill him by poisoning his food. For this reason, Gödel would only eat his wife's cooking. Eventually, he essentially starved himself to death and died at the age of 72 in Princeton.

TURING MACHINE

Turing was born in London in 1912. His father was in the Indian Civil Service and his parents travelled between India and the UK leaving their two sons with friends in England. After schooling, Turing joined the progressive intellectual life

at Cambridge, studied mathematics and was elected a Fellow of King's College in 1935 (aged 23). He then decided to work in mathematical logic and reformulated Gödel's work and in 1936, at 24, devised a computation model called the Turing Machine. Turing proved that the Turing Machine would be capable of performing any conceivable computation.

The 1936 paper gave a definition of computation and an absolute limitation on what computation could achieve, which makes it the founding work of modern Computer Science. It led him to Princeton for more advanced work in logic and he completed a PhD. dissertation there under the supervision of Alonzo Church. On return to England, Turing was engaged in mastering the German enciphering machine, Enigma, from 1939 to 1945. He later went back to Cambridge and then to the University of Manchester in 1949. In 1950, he published his other well-known paper on artificial intelligence and the Turing Test. Turing was arrested in February 1952 for a sexual affair with a young Manchester man, and he was obliged, to escape imprisonment, to undergo the injection of oestrogen intended to negate his sexual drive. He committed suicide in 1954, at his home, at the age of 42.

The Turing Machine has proven itself to be the right theoretical model for computation and the impact of the paper continues to this day. The *Entscheidungsproblem* was proposed by the Hilbert school of mathematicians as the central problem of logic. It dealt with deciding whether a statement in logic followed from given axioms. Turing showed that this was an uncomputable problem.

STORED PROGRAM COMPUTER

The idea of storing the program itself in the memory of the computer and the concept of treating the program as data that could be modified was revolutionary. It was responsible for the computer becoming a tool that is much more valuable than a computer reading instructions from an input device.

There is some controversy about who came up with the stored program idea first but John von Neumann is often credited with it since he wrote the well-known draft report on EDVAC which includes the idea. In addition to discussing the idea, the building of ENIAC and EDVAC at the University of Pennsylvania can be discussed. Some information about Eckert, Mauchly and von Neumann should also be included.

Von Neumann received his doctorate in mathematics from the University of Budapest, in 1926. He lectured at Berlin and then at Hamburg before moving to Princeton in 1930 and becoming a professor there in 1931 at the age of 28. Like Gödel and Turing, von Neumann was an intellectual giant. He was not only a pioneer of the modern digital computer, he made many other contributions including the application of operator theory to quantum mechanics, making a significant contribution to the Manhattan Project, and developing game theory and the concept of cellular automata. Working with Teller and Ulam, von Neumann worked out key steps in the thermonuclear reactions and he himself calculated the precise altitude for the two nuclear weapons detonated above Hiroshima and Nagasaki, to produce the most extensive damage. He died in 1957

(53 years old) of bone cancer and pancreatic cancer possibly contracted through exposure to the radiation of the nuclear tests conducted in 1946.

MEMEX – HYPERTEXT

In a 1945, a paper published in *The Atlantic Monthly*, Vannevar Bush described a microfilm based machine (called Memex for memory extension) that he thought could work something like the way humans think. For example, he suggested it should be able to search by association rather than by indexing. Memex was to have very large storage. Bush noted that "if the user inserted 5000 pages a day, it would take hundreds of years to fill it." Bush further described how numerous documents could be joined together to form a trail and an item could be joined in many trails. These trails could then be saved by a code and recalled by using the code. It took twenty years to pass before the word hypertext was coined by Ted Nelson in his 1965 article.

Bush had an outstanding career, starting with doctorate degrees from Harvard and MIT followed by appointments in Electrical Engineering at MIT. He got involved in constructing a Differential Analyser there and rose to the position of Vice-President and Dean. It appears that he was thinking about Memex in the 1930s. He moved to Washington in 1939 and in 1940, he became chairman of the National Defense Research Committee (NDRC). In 1941 the NDRC was subsumed into the Office of Scientific Research and Development with Bush as director, which controlled the Manhattan Project. In 1945, he recommended the creation of what became the National Science Foundation. One of his PhD students at MIT was Claude Shannon.

TRANSISTORS AND INTEGRATED CIRCUITS

In the early 20th century, long distance telephone signals were amplified along the line using vacuum tubes but the tubes used too much power, produced too much heat and were very unreliable. After the war, in 1945, AT&T decided to find a substitute for the vacuum tubes. A team was set up with William Shockley as the team leader. The close-knit team consisted of John Bardeen (PhD Princeton, theoretical physicist), Walter Brattain (PhD Minnesota, experimental physicist), Shockley (PhD MIT, theoretical physicist) and other physicists, chemists and engineers.

After many unsuccessful attempts to build a semiconductor device, tensions started building in the team and Shockley started working alone. In 1947, Bardeen and Brattain, without telling Shockley, built a point-contact transistor from strips of gold foil on a plastic triangle, pushed down into contact with a slab of germanium. On learning of the invention, Shockley was pleased with the success but furious that he was not involved. He built another type of transistor which was easier to build, pushed Bardeen and Bratton aside, and tried to patent the idea himself. The three broke up in 1948 due to Shockley's abrasive and paranoid management style. Bardeen, Brattain and Shockley won a Nobel Prize for the invention in 1956.

Shockley left Bell Labs when he was not promoted and joined Caltech. With support from a friend he set up Shockley Semiconductors in Palo Alto. He hired superb people although none from Bell Labs would work for him. In 1957, eight of Shockley's top researchers left him and started Fairchild Semiconductor.

Robert Noyce and Gordon Moore then left Fairchild to set up Intel. Others set up National Semiconductor and Advanced Micro Devices.

Shockley's company finally folded and he joined Stanford where he started talking about the least competent people reproducing fastest and the best having fewer children. He thought African Americans had on average 15 points lower IQ than the US population. He suggested voluntary sterilisation for low IQ people. This destroyed his reputation. He died in 1989, aged 79.

Jack Kilby in 1956 created the first integrated circuit to prove that resistors and capacitors could exist on the same piece of semiconductor material. The invention of the transistor and advances in solid state physics led to the development of integrated circuits. These components were smaller, faster and more reliable and consumed less energy. Kilby won a Nobel Prize for his invention in 2000, 44 years after the invention.

INFORMATION THEORY

Claude Shannon went to MIT for graduate studies and completed a Masters thesis "A Symbolic Analysis of Relay and Switching Circuits" in 1937 (aged 21) which was called "possibly the most important Masters thesis of the century"! He completed a PhD under the supervision of Vannevar Bush in 1940.

In 1948, Shannon presented the classical papers on theory of communication. He provided a mathematical definition of information and defined entropy. Shannon described how to measure information as if it was a physical quantity. He also described how optimal codes may be developed for transmitting information. Shannon uses the term entropy in the paper as a substitute for information content. The story goes that Shannon did not know what to call his measure so he asked von Neumann, who said "You should call it entropy ... [since] ... no one knows what entropy really is, so in a debate you will always have the advantage".

Shannon also developed a theory of cryptography in 1949. He wrote a paper on chess playing in 1950 and worked with John McCarthy on artificial intelligence. Shannon was appointed to an Endowed Chair at MIT in 1956. He used his theory successfully in gambling in Las Vegas and on stock market. He loved juggling and built a number of mechanical clowns that could juggle. Shannon died of Alzheimer's disease in 2001, aged 84.

CONCLUDING REMARKS ON THIS APPROACH

Remaining topics from the list of 25 should be covered in a similar way although for some topics the resources available are limited.

For assessment, some essay-type assignments should be used although these may require plagiarism detection to ensure that the essays have not been produced using "cut and paste" materials from the Web. In addition, an examination is recommended to ensure that students have learned some of the basic concepts.

4. Two other possible approaches

We now present two other approaches that may be successfully used for presenting this material to students.

1. Integrate with other courses – this is the approach which has been adopted by recent computing curriculum recommendations. For example, Computing Curriculum 2001 includes history of computer architecture, operating systems, networking, wireless, programming languages, artificial intelligence, information systems, database systems, hypertext as well as history of Computer Science in the curriculum recommendations. The thematic list is easily used for the integration approach and it has a number of advantages since the ideas are presented in the context of the relevant topic. As an example, the two lists below may be used for inclusion in courses on programming languages and on theory respectively.

   Programming Languages

   - Fortran and its Compiler
   - Algol 60 Definition and BNF
   - Lisp and Functional Programming
   - Simula and Object-oriented Programming

   Theory

   - Incompleteness
   - Decidability and Turing Machines
   - Information Theory, Entropy
   - Computational Complexity
   - NP-Completeness

2. As a separate thematic course – a thematic presentation based on lists like the two given above works quite well. This approach has been used when the material has been presented in a series of seminars. It helps relate ideas in one CS topic to material that the students may have already covered in other courses. One minor weakness of this approach however appears to be that too much emphasis is often placed on the themes and discussion often gets focused on important ideas in various themes. This can however be carefully managed and the approach can lead to interesting class discussions.

A course about revolutionary ideas can also be taught as one of the first courses in a graduate program as this material provides an historical perspective on the most important ideas in Computer Science that should be motivating for new graduate students.

5. Concluding remarks

I have presented a proposal for a course on 25 revolutionary CS ideas of the 20th century. This proposal is quite different from proposals for computing history discussed earlier. This course should be included in the CS curriculum at the final year level. The course is likely to be useful as an introductory graduate course. In addition to looking at CS history through the "lens" of the revolutionary ideas, the course should motivate students to do graduate work.

As noted earlier, the list presented is a subjective personal list of ideas and I expect readers to disagree with some of the selections but hope they will agree with most of them. Teachers should use their own personal list in teaching the course so that they feel ownership of the course curriculum.

The ideas listed here were presented in a course on history of computing and professional and social issues using the approach presented in Section 3. The enthusiasm of students in the course was surprising. A collection of Powerpoint slides used in this course is available at http://www.csse.monash.edu.au/~gopal.

APPENDIX – PRIMARY RESOURCE MATERIAL

K. Gödel, *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*, Dover, Reprint edition, 80pp, 1992.

A.M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society, 1936, Ser. 2, Vol. 42, pp. 230-265; Vol. 43, pp. 544-546

V. Bush, As We May Think, *The Atlantic Monthly*, July 1945.

J. von Neumann, First Draft of a Report on the EDVAC, Moore School of Electrical Engineering, Univ. of Pennsylvania, Philadelphia, June 30, 1945. (Also published in *IEEE Annals of the History of Computing*, Vol. 15, No. 4, pp. 27-75, 1993.)

J. Bardeen and W.H. Brattain, Three-Electrode Circuit Element Utilizing Semiconductive Materials, US Patent 2524035, Filed 1948, Granted 1950.

C. Shannon, A Mathematical Theory of Communication, Bell System Tech. J., 1948, Vol. 27, pp. 379-423, 623-656, July, October 1948.

A.M. Turing, Computing Machinery and Intelligence, *Mind*, 59, 1950, 433-60 Also in Perspectives on the Computer Revolution, pp. 224-245.

W.W. Peterson, Addressing for Random-Access Storage, *IBM Journal of Research and Development*, Vol. 1, No. 2, pp. 130-146, 1957.

J.W. Backus *et al*, The FORTRAN automatic coding system. In Proceedings of the Western Joint Computer Conference, pp. 188-198, February 1957.

J. McCarthy, Recursive Functions of Symbolic Expressions and their Computation by Machine, Part I. *Communications of the ACM*, Vol. 3, 1960, pp. 184-195.

P. Naur *et al* (Eds), Revised Report on the Algorithmic Language Algol 60, *Communications of the ACM*, Vol. 6, No. 1, January 1963, pp 1-17.

P. Baran. On Distributed Communications Networks. *IEEE Transactions on Communications*, pp. 1-9, 1964.

I.E. Sutherland, Sketchpad: A Man-Machine Graphical Communication System, *Proceedings of the AFIPS Spring Joint Computer Conference*. Washington DC, 1963, pp. 329-346.

G. Amdahl, F. Brooks, and G. A. Blaauw, Architecture of the IBM System/360, *IBM Journal of R & D*, Vol. 8, No. 2, 1964.

J. Hartmanis and R. Stearns, On the computational complexity of algorithms, *Transactions of the American Mathematical Society*, 117 (1965), pp. 285-306.

O.J. Dahl and K. Nygaard, SIMULA – an ALGOL-Based Simulation Language, *Communications of the ACM*, Vol. 9, No. 9, September 1966, pp. 671-678.

A.F. Westin, Legal Safeguards to Insure Privacy in a Computer Society, *Communications of the ACM*, Vol. 10, No. 9, September 1967, pp. 533-537.

E.F. Codd, A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, Vol. 13, No. 6, June 1970, pp. 377-387.

S.A. Cook, The Complexity of Theorem-Proving Procedures, in *Proc. 3rd Ann. ACM Symposium on Theory of Computing*, pp. 151-158. ACM, New York, 1971.

D.M. Ritchie and K. Thompson, The UNIX Time-Sharing System, *Communications of the ACM*, Vol. 17, No. 7, July 1974, pp. 365-375.

C.P. Thacker, E.M. McCreight, B.W. Lampson, R.F. Sproull, and D. R. Boggs. Alto: A Personal Computer. D. Siewiorek, C. G. Bell, and A. Newell (Eds), *Computer Structures Readings and Examples*, editors, second edition, McGraw-Hill, 1979. 20.

R.L. Rivest, A. Shamir and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, February 1978, pp. 120-126.

T. Berners-Lee, Information Management – A proposal, March 1989, May 1990, 20pp.

S. Brin, and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, *Computer Networks and ISDN Systems*, Vol. 30, No. 1-7, pp. 107-117, April 1, 1999.

REFERENCES

1. A.W. Biermann, *Great Ideas in Computer Science* - 2nd Edition: A Gentle Introduction, The MIT Press, 1997.

2. CC 2001 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, Final Report of the Joint ACM/IEEE-CS Task Force on Computing Curricula 2001 for Computer Science.

3. T.J. Cortina and R. McKenna, The design of a history of computing course with a unique perspective, *ACM SIGCSE Bulletin*, Proceedings of the 37th

SIGCSE technical symposium on Computer science education SIGCSE '06, Vol. 38, Issue 1, March 2006.

4. A.K. Dewdney, *The New Turing Omnibus: Sixty-Six Excursions in Computer Science*, Owl Books; Reprint edition, 1993.

5. M. Eisenberg, Creating a computer science canon: a course of "classic" readings in computer science, Proceedings of the 34th SIGCSE technical symposium on computer science education, Reno, Nevada, USA, pp. 336–340, 2003.

6. J. Impagliazzo, M. Campbell-Kelly, G.B. Davis, J.A.N. Lee, and M.R. Williams: IFIP TC3/TC9 Joint Task Group: History in the Computing Curriculum. *IEEE Annals of the History of Computing*, Vol. 21, No. 1, pp. 4-16 (1999).

7. T. Kuhn, *The Structure of Scientific Revolutions*, University of Chicago Press, 1962.

8. P. Laplante, *Great Papers in Computer Science*, West Publishing Company, 1996.

9. J.A.N. Lee, History in the computer science curriculum, *ACM SIGCSE Bulletin*, Vol. 28, Issue 2, June 1996.

10. J.A.N. Lee, History in the computer science curriculum, *ACM SIGCSE Bulletin*, Vol. 29, Issue 4, December 1997.

11. Z.W. Pylyshyn (Ed.), *Perspectives on the Computer Revolution*, Prentice Hall, 1970.