In dieser Armee wird es keine Software geben!
Dwight Eisenhower

Die Entwicklung der Software

Der Begriff «Software» ist bezüglich seiner Herkunft umstritten. Wahrscheinlich wurde er 1958 erstmals vom Statistiker John Wilder Tukey aus Princeton im heutigen Sinne gebraucht. Software wird gemeinhin als Gegenstück zum Sammelbegriff «Hardware» verstanden und für Daten und alle möglichen Arten von Programmen verwendet. Ein Computerprogramm ist die Übersetzung eines Algorithmus in eine für den Computer verständliche Sprache. Eine Verarbeitungsprozedur benötigt in der Regel eine Eingabesprache, einen Übersetzungsalgorithmus und eine vom Computer verstehbare Maschinensprache. Es hat sich daher eingebürgert zwischen System- und Anwendersoftware zu unterscheiden. Zur Systemsoftware gehören das Betriebssystem, Benutzeroberflächen, Programmiersprachen, Kommunikationsprogramme für den Informationsaustausch sowie diverse Hilfsprogramme (utilities), während die Anwenderprogramme oder Applikationen, dem Anwender bei der Lösung spezifischer Probleme helfen wie beispielsweise dem Schreiben eines Textes oder der Erstellung einer Kalkulation.

Das erste Programm

Das erste Programm ist älter als der erste Computer. Als Charles Babbage 1840 an einer Tagung in Turin seine Pläne für eine *Analytical Engine* vorstellte, verfasste ein Mathematiker, welcher später zum Ministerpräsidenten Italiens aufsteigen sollte, darüber einen Bericht, der unter dem Titel, «notions sur la machine analytique», 1842 in Genf erschien. Die Schrift Luigi Menabreas fiel einer mathematikinteressierten jungen Dame in die Hände, welche an Babbages Arbeit an der *Analytical Engine* regen Anteil nahm. Sie beschloss, den Aufsatz ins Englische zu übersetzen, einige Fehler zu korrigieren und ihn mit persönlichen Anmerkungen zu versehen. Am Schluss machten ihre Anmerkungen das Dreifache des ursprünglichen Textes aus und zeigten, am Beispiel der Bernoulli-Zahlen einen Regelkatalog und eine Abfolge von Arbeitsgängen auf, mit welchen Babbages Maschine – so sie denn gebaut würde – nicht nur die Bernoulli-Zahlen berechnen, sondern auch viel allgemeinere Operationen durchführen könnte.

> It may be desirable to explain, that by the word *operation*, we mean *any process which alters the mutual relation of two or more things*, be this relation of what kind it may. This is the most general definition, and would include all subjects in the universe.

Damit hatte die Tochter des Dichters Byron etwas geschaffen, was der Dichtung ihres berühmten Vaters ebenbürtig war: Ada King hatte das erste Programm für einen Rechenautomaten entworfen.

In der Pionierzeit des Computerbaus wurden die Computer für bestimmte Zwecke gebaut und besaßen eine *Verdrahtungsprogrammierung*. Ein Programm umfasste Unmengen von Drähten, die jeweils umgesteckt werden mussten, wenn der Computer „umprogrammiert“ werden sollte. Beim ENIAC, der der Berechnung ballistischer Tafeln diente, benötigten die vorwiegend weiblichen Operators in der Regel zwei Tage, bis die für die Programmsteuerung erforderliche Verkabelung fehlerfrei vorgenommen war. Man suchte daher rasch nach Möglichkeiten, den Computer schneller umzuprogrammieren. Dazu wurde sogenannter Maschinencode entwickelt. Dieser wurde in der Nachfolge von Konrad Zuse binär codiert (1/0), damit ließen sich die

Steuersignale direkt aus der Befehlsliste ableiten. Das Erstellen solcher Programme war zwar einfacher als vorher, aber immer noch umständlich und erfolgte in der Zeit der Großcomputer durch die Erbauer der Anlagen selbst. Dazu wurden die Programmbefehle zunächst in Lochstreifen oder Lochkarten gestanzt und mussten anschließend in den Computer eingelesen werden. Dieses Verfahren war fehleranfällig, entsprechend langwierig und teuer.
Schon John P. Eckert und John W. Mauchly träumten von einem Computer, welcher nicht nur Daten, sondern auch Programme speichern konnte. Dieses Konzept, das heute mit dem Namen des ungarischen Mathematikers John von Neumann verbunden wird, stellte den ersten Schritt in Richtung moderne Programmiersprachen dar. Verwirklicht wurde es erstmals 1949 im EDSAC (= Electronic Delay Storage Automatic Computer) von Maurice Wilkes an der Universität Cambridge (GB).
Die Rechner konnten nun umfangreichere Berechnungen ausführen, aber im Gegenzug erhöhte sich der Programmieraufwand und die Wahrscheinlichkeit, falsch zu programmieren. Deshalb wurde nach Lösungen gesucht, das Programmieren zu vereinfachen, zum Beispiel, indem man gewissen Folgen von Einsen und Nullen eine Taste des Fernschreibers zuordnete oder ein Programm entwickelte, das mathematische Gleichungen in Originalschreibweise verstand.
Für den sogenannt mnemotechnischen Code wurden die Bit-Muster in bedeutungstragende Buchstabenfolgen umgewandelt (z.B. 0001 = ADD). Die Übersetzung des Codes in die Bitmuster des Binärcodes erfolgte nun, einer Idee von Heinz Rutishauser folgend, nicht mehr mit einem separaten Gerät, sondern mittels eines Assemblers auf der Rechenmaschine selber. Assembler waren aber noch an die Hardware gebunden, das heisst, sie liefen nicht auf Computern anderer Hersteller.

Den nächsten Entwicklungsschritt stellten die problemorientierten Sprachen dar, in welchen unabhängig vom System, auf dem das Programm laufen sollte, programmiert werden konnte. Für die automatische Übersetzung von Programm- in Maschinencode waren nun, dank der Überlegungen von Grace Hopper, sogenannte Compiler zuständig, die der Hersteller des Rechners lieferte. Solche mechanischen Verfahren waren deutlich weniger fehleranfällig als das Coden von Hand.

Höhere Programmiersprachen

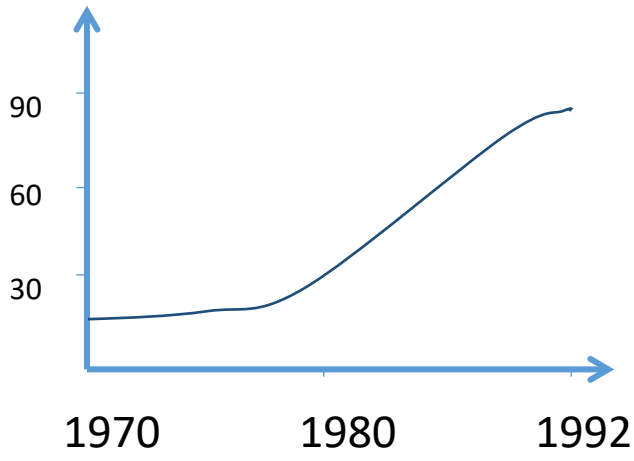
Am Beginn der Entwicklung der höheren Programmiersprachen standen – neben Zuses einsamen, weil ohne Compiler dastehendem Plankalkül von 1945/46 – FORTRAN und das aus Flow-Matic hervorgegangene COBOL. Beide abstrahierten von maschinenspezifischen Details und konnten im Idealfall auf verschiedenen Rechensystemen eingesetzt werden.
Die Übertragung von Programmen auf andere Systeme blieb aber noch für lange Zeit schwierig. FORTRAN bedeutet „Formula Translator“ und geht auf John Backus (1924–2007) zurück, welcher für die Labors von IBM tätig war. Diese Sprache war auf die numerischen Wissenschaften und Großcomputer ausgerichtet. Die Sprache COBOL ist das Kind der Erfinderin des Compilers, Grace Hopper (1906–1992), und wurde für wirtschaftliche Anwendungen und kleinere Computer konzipiert. Sie lässt sich beinah wie Englisch lesen und verdankt ihren Erfolg zu einem großen Teil der Tatsache, dass die amerikanische Regierung die Installation eines kommerziellen Computers vom Vorhandensein eines COBOL-Compilers abhängig machte. Zwischen FORTRAN und COBOL steht ALGOL, eine Sprache, die als universell nutzbare, maschinenunabhängige, algebraische Sprache für wissenschaftliche Zwecke gedacht war. Wesentliche Beiträge zu dieser Sprache lieferten der oben erwähnte Heinz Rutishauser (1918–1970) von der ETH in Zürich, Friedrich L. Bauer (1924–2015) in Mainz und John Backus (1924–2007) von IBM. Nach anfänglichen Schwierigkeiten wurde ALGOL 1960 auf

einer Konferenz in Paris überarbeitet und schien auf dem Weg das «Latein der Programmiersprachen» zu werden. Die weitere Entwicklung dieser Sprache unter Adriaan van Wijngaarden (1916–1987) mutierte jedoch zu einem «Mammutunternehmen» und wird mehrheitlich als gescheitert betrachtet. Sie inspirierte aber die Entwicklung weiterer Sprachen unter anderem von PASCAL, MODULA und OBERON von Niklaus Wirth (1934–).

```
BEGIN
FILE F (KIND=REMOTE);
EBCDIC ARRAY E [Ø:11];
REPALCE E BY «HELLO WORLD!»;
WHILE TRUE DO
    BEGIN
    WRITE (F, *, E);
    END
END.
```

ALGOL Programm-Code

Eine vierte wichtige Sprache aus den Anfängen der höheren Programmiersprachen am Ende der fünfziger Jahre ist LISP, die auf John McCarthy (1927–2011) zurückgeht. Diese Sprache arbeitete mit nichtnumerischen Symbolen, besaß eine einfache syntaktische Struktur und war auf Kreativität ausgerichtet. Sie fand ihre Anhänger im Umfeld der KI-Forschung in Stanford und am MIT, wo McCarthy tätig war, nicht aber in der Ausbildung und der Industrie. Trotz ihrer Beschränkung auf Hochschulen hat sie auf die Entwicklung der Informatik großen Einfluss ausgeübt. Aus der LISP-Kultur ergab sich zum Beispiel die heute gängige Form der Mensch-Maschine-Interaktion.
Nach 1960 entstanden rasch weitere Sprachen. 1961 zählte man bereits über 70. Trotz dieses Wachstums hinkte die Programmierung den Fortschritten in der Hardware-Entwicklung hinterher: Die Maschinen wurden rascher potenter als die Programmiersprachen besser. Laufzeit und Platzverbrauch des Programms spielten zunehmend eine wichtigere Rolle. Die Folgen davon waren, dass der sogenannte «Spaghetti-Code» wucherte und die Software teurer wurde als die Hardware. Im Oktober 1968 lud die NATO, welche von allfälligen Programmfehlern besonders betroffen gewesen wäre, zu einer Tagung in Garmisch-Partenkirchen ein, um die «Software-Krise», wie sie Edsger Dijkstra (1930–2002) rückblickend nannte, zu überwinden, indem man das Programmieren strukturiert betrieb.

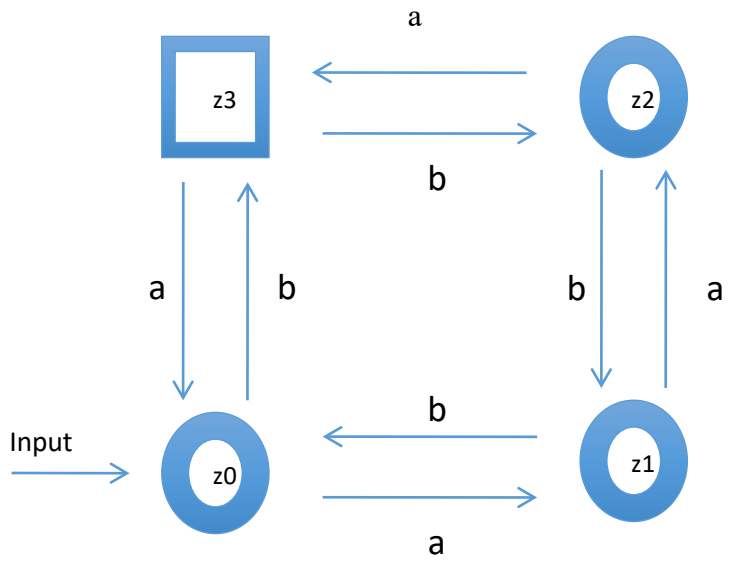


Anteil der Software an den Kosten von Computersystemen

In Garmisch und ein Jahr später in Rom wurde die Forderung erhoben, die Software-Entwicklung auf eine theoretische Grundlage zu stellen und ingenieurmäßig zu betreiben. Im gleichen Jahr veröffentlichte Donald Knuth (1938–), welcher sich seit anfangs der 60er Jahre der systematischen Untersuchung nichtnumerischer Algorithmen widmete, den ersten Band von «The Art of Computer Programming». Und ebenfalls 1968 beschloss IBM als Folge eines Kartellverfahrens, Hard- und Software zu entflechten (unbundle), was soviel bedeutete wie getrennt zu verkaufen. Weitere wichtige Promotoren der Entwicklung des *Software Engineering* waren: Friedrich Bauer, der Vater des Begriffs Software Engineering, David Lorge Parnas (1941–) und Barry Boehm (1935–).Boehm und Parnas unterteilten die systematische Software-Entwicklung in verschiedene, aufeinander folgende Phasen, die es iterativ zu durchlaufen gilt, wenn man stabile, zweckdienliche und erschwingliche Software herstellen will. Nach Rechenberg unterscheidet man in der Regel eine Phase der Problemanalyse, eine des Entwurfs, eine der Implementierung, des Testens und der Wartung. Ein ganz zentrales Anliegen von Parnas war und ist die saubere Dokumentation von Software.

Ein Jahr nach Garmisch begannen Ken Thompsen (1943–) und Dennis Ritchie (1941–2011) in den Bell Laboratories in New Jersey mit der Arbeit an dem, woraus das Betriebssystem UNIX und die Programmiersprache C werden sollte. Ihr Motto, «keep it simple, stupid», ist als Akronym KISS auch außerhalb der Informatik zu einem verbreiteten Prinzip geworden. Etwa gleichzeitig machte sich Niklaus Wirth, der C für einen «leap backward» hielt, in Zürich im Alleingang an den Entwurf einer einfachen, für die Ausbildung von Informatikern geeigneten Programmiersprache, die er zu Ehren des großen französischen Mathematikers «Pascal» nannte. Damit stieg das Software-Engineering endgültig vom Handwerk zur akademischen Disziplin empor, die dem Bau von Hardware ebenbürtig war: Die Informatik im modernen Sinn war geboren.

Parallel zur Entwicklung der Programmiersprachen entstanden ab 1955 auch die ersten reinen Software-Firmen und neue Teildisziplinen der Informatik, zum Beispiel die Theorie der formalen Sprachen, der *Automatentheorie* oder später das *Requirements Engineering*.
Die erste Disziplin hat ihren Anfang im Jahre 1957 mit Noam Chomskys «Syntactic Structures» genommen. Der Linguist Chomsky (1928–) verfolgte zwei Anliegen: Die Suche nach universellen Regeln der Grammatik und deren mathematisch genaue Beschreibung, die es uns ermöglichen, die Richtigkeit von Sätzen zu beurteilen, auch wenn wir sie noch nie gehört haben. Obwohl Chomsky von natürlichen Sprachen ausging, konnten seine Erkenntnisse auch für die formale Beschreibung von Computersprachen genutzt werden; denn auch diese bestehen aus strukturierten Zeichenketten, genau wie sprachliche Texte. Chomsky entwickelte eine Typologie, mit der sich Grammatiken in vier Klassen einteilen lassen: Phrasenstrukturgrammatiken (Typ-0-Grammatiken), kontextsensitive Grammatiken (Typ-1-Grammatiken), kontextfreie Grammatiken (Typ-2-Grammatiken), reguläre Grammatiken (Typ-3-Grammatiken). Nach Chomsky ist eine Sprache L eine Typ-n-Sprache, wenn eine Typ-n-Grammatik existiert, welche die Sprache L erzeugt.
Chomskys Arbeit hatte auch Einfluss auf die *Automatentheorie*. Denn Automaten im Sinne der *Theoretischen Informatik* dienen der Erzeugung oder Erkennung von Sprachen. Diese werden wie die Sprachen in vier Kategorien eingeteilt, in endliche Automaten (welche reguläre Sprachen akzeptieren (Typ-3); Kellerautomaten, die einen Speicher haben und kontextfreie Sprachen akzeptieren (Typ-2); linear beschränkte Automaten mit einem endlichen Band für kontextsensitive Sprachen (Typ-1) und Turing-Maschinen mit einem unendlichen Band für dieTyp-0-Grammatiken.



Endlicher Automat für Typ-3-Sprachen

Die letzte Disziplin, das *Requirements Engineering* (RE) oder die Anforderungsanalyse, steht am Anfang jedes erfolgreichen Entwicklungsprozesses und nimmt deshalb eine zentrale Rolle in der Informatik ein. Der Begriff wurde 1979 erstmals verwendet, ist aber erst 1990 allgemein in Gebrauch gekommen. Mitte der 90er Jahre, als sich die Informatik dank des Internets rasant in viele neue Bereiche ausbreitete, hat sich das RE als Teildisziplin der Informatik etabliert, und es gibt seither Zeitschriften, Tagungen und Lehrstühle zu diesem Fachgebiet.

Standardsoftware

Anfangs der 1970er Jahre waren die Computer dialogfähig und dank der Serienfertigung wesentlich kleiner und billiger, so dass auch kleinere Firmen und Privatpersonen Computer erwerben konnten. Aber ein sogenannter «Minicomputer» wie der PDP-8 der Firma DEC hatte Abmessungen von 60 x 60 x 180 Centimeter, war 110 Kilo schwer und kostete immer noch 18 000 Dollar. Erst die Chiptechnologie der Firma Intel verkleinerte Mitte der 1970er Jahre die Computer soweit, dass man wirklich von einem Heim- oder Mikrocomputer, wie sie zunächst hiessen, sprechen konnte. Es stellte sich jedoch die Frage, wie die neue Benutzergruppe zu Programmen für ihre gekauften Mikrocomputer kam. Die ersten Heimcomputer wie der Altair (1975) oder auch der Commodore PET (1977) fanden noch ohne Anwenderprogramme Käufer, da diese selber in BASIC programmieren konnten. BASIC, das für «**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode» steht, war 1964 von John Kemeny und Thomas Kurtz am Dartmouth College entwickelt worden als Sprache für das Erlernen des Programmierens. Es war einfacher zu erlernen als ALGOL oder FORTRAN, war aber wenig elaboriert, arbeitete es doch noch mit Zeilennummern und Sprungbefehlen (GOTO), welche in den Reihen der akademischen Softwareentwickler als eher schädlich betrachtet wurden. Wesentlichen Anteil an der Verbreitung von BASIC hatten Paul Allen und Bill Gates. Sie hatten in Seattle die gleiche Schule besucht und sich dort erste Programmierkenntnisse angeeignet. Als sie in der Zeitschrift «Popular Electronics» vom Intel-Chip 8080 gelesen hatten, erkannten sie – schneller als die grossen Computerfirmen – dessen Potenzial für die Herstellung von Software für Mikrocomputer für sich abzeichnenden einen Massenmarkt. Sie gründeten eine Firma und lancierten eine abgespeckte BASIC-Variante für den Altair, die nur 4K Speicher brauchte und

so erfolgreich verkauft werden konnte, dass ihre Firma 20 Jahre später 17 000 Angestellte zählte, und der Name «Microsoft» bis heute zu den bekanntesten Brands der Welt gehört.
Mit der Verbreitung von Mikrocomputern entstand in den 1980er Jahren ein rasch wachsendes Bedürfnis nach spezifischen Anwendungen wie Schreibprogramme oder Tabellenkalkulation. Die Firma Commodore löste das Problem in der Schweiz zum Beispiel so, dass sie Schulen wie zum Beispiel der HTL Brugg-Windisch Computer schenkte unter der Bedingung, dass die für diese Computer hergestellten Applikationen von ihr weitervertrieben werden durften.
Andere Firmen wie IBM vergaben im Rahmen des sogenannten *Unbundling* Aufträge zur Software-Entwicklung an kleinere Firmen wie etwa die Firma Microsoft des oben erwähnten Bill Gates, dessen Dienste zeitweilig auch Steve Jobs von Apple in Anspruch nahm, weil Gates – im Gegensatz zu Jobs – anständig BASIC programmieren konnte. Die Zusammenarbeit der Grossfirma IBM mit Microsoft im Bereich der Mikrocomputer führte dazu, dass IBM und MSDOS über Jahre hinweg zu einem Quasistandard wurden. Weil Architektur, Prozessor und Betriebssystem frei erwerbbar waren, war es leicht, kompatible Maschinen herzustellen, und mit zunehmender Menge an kompatiblen Maschinen wuchs der Anreiz für Applikationsentwickler, Programme für sie herzustellen. Die Käufer wiederum wollten diejenige Maschine, für die am meisten Software vorhanden war, weshalb innerhalb weniger Jahre praktisch alle konkurrierenden Betriebssystem-Standards bei den sogenannten «Personalcomputern» verschwanden.
Eine dritte Variante, Applikationen für seine Computer zu erhalten, bestand darin, dass sich Firmen ein Programmiererteam ins Haus holten und diesem erlaubten, die entwickelten Programme an weitere Kunden zu verkaufen. So entstand 1972 aus fünf Mitarbeitern von IBM Deutschland die Firma SAP.

Proprietäre versus Freie Software

Daraus, dass Mitte der 1980er Jahre aus dem Verkauf von Software ein Massenmarkt wurde, ergaben sich auch Probleme, zunächst vor allem für die Grossen der Branche. Diese fürchteten einerseits die IBM-Klone unter MS DOS und andererseits die graphische Benutzeroberfläche von Apple. Deshalb gründeten sie 1988 die Open Software Foundation, um die Vereinheitlichung von UNIX zu fördern, für das seit 1975 auch kommerzielle Lizenzen erhältlich waren. Das angestrebte Quasimonopol der Firmen, IBM AT&T, Bull, HP, IBM, DEC, Nixdorf, Olivetti, Philips und Siemens, kam jedoch nicht zustande: Zum einen wegen der Uneinigkeit der beteiligten Firmen, vor allem aber wegen der Hackerkultur an den amerikanischen Hochschulen. Hacker beschreibt Steven Levy in seinem berühmt gewordenen Text als Leute um den legendären TX-o am MIT, die geprägt waren von der Idee der Freiheit sowie einem Misstrauen gegenüber Autoritäten und Bürokratie. Sie ärgerten sich darüber, dass private Firmen begannen, von ihnen in Forschungslabors entwickelte Programme, die gemäss dem Prinzip des *common law* der Allgemeinheit gehören sollten, weiterentwickelten und anschliessend verkauften. Aus ihrem Unbehagen entstand 1984 das sogenannte GNU-Projekt von Richard Stallman (1953–).Stallman soll sich daran gestört haben, dass Xerox den Quellcode für einen Drucker in seinem Labor nicht offenlegen wollte, und beschloss, selbst ein Betriebssystem samt Compiler und Texteditor zu entwerfen nach dem Vorbild von UNIX. Der Name GNU steht denn auch für *GNU is Not Unix.*
Um zu verhindern, dass dieses System von einer Firma adaptiert und als proprietäres System verkauft werden könnte, wie UNIX System V von AT&T, kündigte er seine Stelle am MIT und unterstellte sein Betriebssystem der *General Public License* (GPL). Diese besagt, dass jedermann die Software gratis nutzen, weiterentwickeln und weitergeben darf. Das bedeutet,

dass man den Quellcode offenlegen muss, keine Lizenzgebühren erheben darf und verbesserte oder neue Programme ebenfalls unter der GPL lizenzieren muss.
Eric Raymond (1957–), welcher wie Stallman aus der Hackerszene stammt, hat in seinem berühmten Aufsatz «The Cathedral and the Bazaar» dargelegt, wie sich die Software-Entwicklung in den Freiwilligenteams der Hackerkultur (Bazaar) von derjenigen in Softwarefirmen (Cathedral) unterschied. Die – größtenteils unausgesprochenen – Verhaltensregeln der Hackerkultur basieren auf dem Grundsatz, dass Autorität die Folge von Fach- und Sozial-Kompetenz ist. Sein Aufsatz ist auch heute noch lesenswert.

Mit dem Kernel von LINUX war 1994 der Weg frei für den Einzug der freien Software auf den *Personal Computern*. Dank der verbesserten Kommunikationsmöglichkeiten durch das Internet entstand unter der Führung von Eric Raymond, Bruce Perens und dem Verleger Tim O'Reilly die *Open-Source-Bewegung*, welche bereit war, mit der Industrie zusammenzuarbeiten, und dies im Falle von Netscape auch erfolgreich tat.
Berühmte Beispiele für Software, die als *freie Software* unter der GNU-Lizenz entstanden, sind die Textverarbeitungssoftware *Starwriter* von Marco Börries (1968–), das Betriebssystem LINUX von Linus Torvalds (1969–), PHP von Rasmus Lerdorf (1968–), die Graphiksoftware GIMP von Peter Mattis und Spencer Kimball sowie der Datenbankserver MySQL von Michael Widenius (1962–). Immer häufiger kam es auch vor, dass Firmen ihren Quellcode freigaben, wie beispielsweise *Netscape* mit dem Mozilla-Browser oder *Sun* mit dem Betriebssystem *Solaris*. Am Beispiel von IBM, das 2001 mit viel PR-Aufwand verbunden erklärte, LINUX als Referenzsystem für alle künftigen Entwicklungen zu machen, zeigt sich jedoch exemplarisch eine Doppelmoral gewisser Firmen; denn IBM bedient sich zwar bei permissiv lizenzierten Projekten, lizenziert selber aber nicht unter GPL. Im Oktober 2018 gab IBM bekannt, dass sie die derzeit grösste Open-Source-Firma der Welt, die 1993 gegründete *Red Hat*, kaufen wolle. *Red Hat* hatte bis dahin ausschliesslich davon gelebt, dass sie ihr Betriebssystem gratis abgab und nur für Unterhalt und Support Rechnung stellte. IBM, das der Börsenentwicklung etwas hinterher hinkte, bot für den grössten Zukauf seiner Geschichte 34 Milliarden Dollar.
Ähnliches gilt auch von der 1996 gegründeten Firma *Google*. Richard Stallman, den Vater von GPL, freute die Popularisierung von LINUX und *Firefox* deshalb nur bedingt, weil er durch *Open Source* die politischen Aspekte der GPL und die Verdienste von GNU in den Hintergrund treten sah.
Mengenmässig hat sich LINUX gegenüber Microsoft bisher nur schleppend etabliert. Im Jahr 2009 hatten die Betriebssysteme von Microsoft weltweit einen Marktanteil von über 90 Prozent, im Juli 2014 waren es immer noch über 80 Prozent. Ein Grund für die marktbeherrschende Stellung von Microsoft könnte der Umstand sein, dass Windows in der Regel zusammen mit dem Internet-Browser und dem Office-Paket angeboten wird und einen Quasistandard bildet. In jüngster Zeit gab es jedoch – nicht zuletzt bedingt durch die Enthüllungen des amerikanischen Whistle-Blowers Edward Snowden (1983–) – wieder vermehrt Tendenzen zu einer Abkehr von Microsoft. 2014 leitete beispielsweise die chinesische Regierung Schritte in diese Richtung ein.

Literatur

Bachmann, Manuel (2014): Triumph des Algorithmus. wie die Idee der Software erfunden wurde. Zürich: Swiss Engineering Institute Press

Balzert, Helmut (2001): Lehrbuch Software-Entwicklung. Heidelberg: Springer

Bauer, Friedrich, L. (2007): Kurze Geschichte der Informatik. München: Fink

Brügge, Bern/Harhoff, Dietmar/Picot, Arnold/Creighton, Oliver/Fiedler, Mrina/Henkel, Joachim (2004): Open-Source-Software. Eine ökonomische und technische Analyse. Berlin, Heidelberg, New York: Springer

Ceruzzi, Paul E. (2003): Eine kleine Geschichte der EDV. Bonn: MITP

Chomsky, Avram Noam (1957): Syntactic Structures. The Hague, Paris: Mouton

Conlan, Roberta; Hrsg. (1990): Software. Amsterdam: Time-Life Books

Dijkstra, Edsger W. (1972): The Humble Programmer. Communications oft ACM. ACM 15/10, S. 859–866

Furger, Franco (1993): Informatikinnovationen aus der Schweiz. Lilith/Diser und Oberon. Zürich: Technopark

Gates, Bill (1995): Der Weg nach vorn. Hamburg: Hoffmann & Campe

Gleick, James (2011): Die Information. München: Redline

Gruntz, Dominik (2011): Einführung in Java. Vorlesungsfolie. Windisch: Hochschule für Technik FHNW

Henger, Gregor (2008): Informatik in der Schweiz. Zürich: NZZ

Imhorst, Christian (2004): Die Anarchie der Hacker. Marburg: Tectum

Levy, Steven (1984): Hackers. Heroes of the Computer Revolution. New York: Duobleday

Menabrea, Luigi (1842): Notions sur la machine analytique de M. Charles Babbage. Genève: Bibliothèque Universelle de Genève 82/10, S. 325–376

Moravec, Hans (1990): Mind Children. Hamburg: Hoffmann und Campe

Naumann, Friedrich (2001): Vom Abakus zum Internet. Die Geschichte der Informatik. Darmstadt: WBG

Raymond, Eric: The Cathedral & the Bazaar. Vgl. www.catb.org/~esr/writings/homesteading/homesteading/ (30.5.2013)

O'Reagan, Gerard (2008): A Brief History of Computing. London: Springer

Parnas, David L. (2011): Precise Documentation: The Key to better Software. In: Nanz, Sebastian; Hrsg. (2011): The Future of Software Eningeering. Heidelberg: Springer, S. 125–148

Rechenberg, Peter (2000): Was ist Informatik? München: Hanser

Siefkes, Dirk e.a. Hrsg. (1999): Pioniere der Informatik. Berlin: Springer

Sketch of the Analytical Engine Invented by Charles Babbage. By L. F. Menabrea of Turin, Officer of the Military Engineers, from the Bibliothèque Universelle de Genève, October, 1842, No. 82. With notes upon the Memoir by the Translator Ada Augusta, Countess of Lovelace. Vgl. http://www.fourmilab.ch/babbage/sketch.html (21.8.2012)

Spiegel, André (2006): Die Befreiung der Information. GNU, Linux und die Folgen. Berlin: Matthes & Seitz

Thayer, R. H. / Dorfmann, M. eds (1990): System and Software Requirements Engineering. Los Alamos: IEEE Computer Society Press

Weiss, Robert (1993): Mit dem Computer auf „DU“. Männedorf: Weiss

Wexelblat, Richard L. (1981): History of Programming Languages. New York: ACM

Wieland, H.R. (2011): Computergeschichte(n) – nicht nur für Geeks. Bonn: Galileo

Wirth, Niklaus (2008): A Brief History of Software Engineering, Vgl: http://www.inf.ethz.ch/personal/wirth/Articles/Miscellaneous/IEEE-Annals.pdf (3.9.2013)

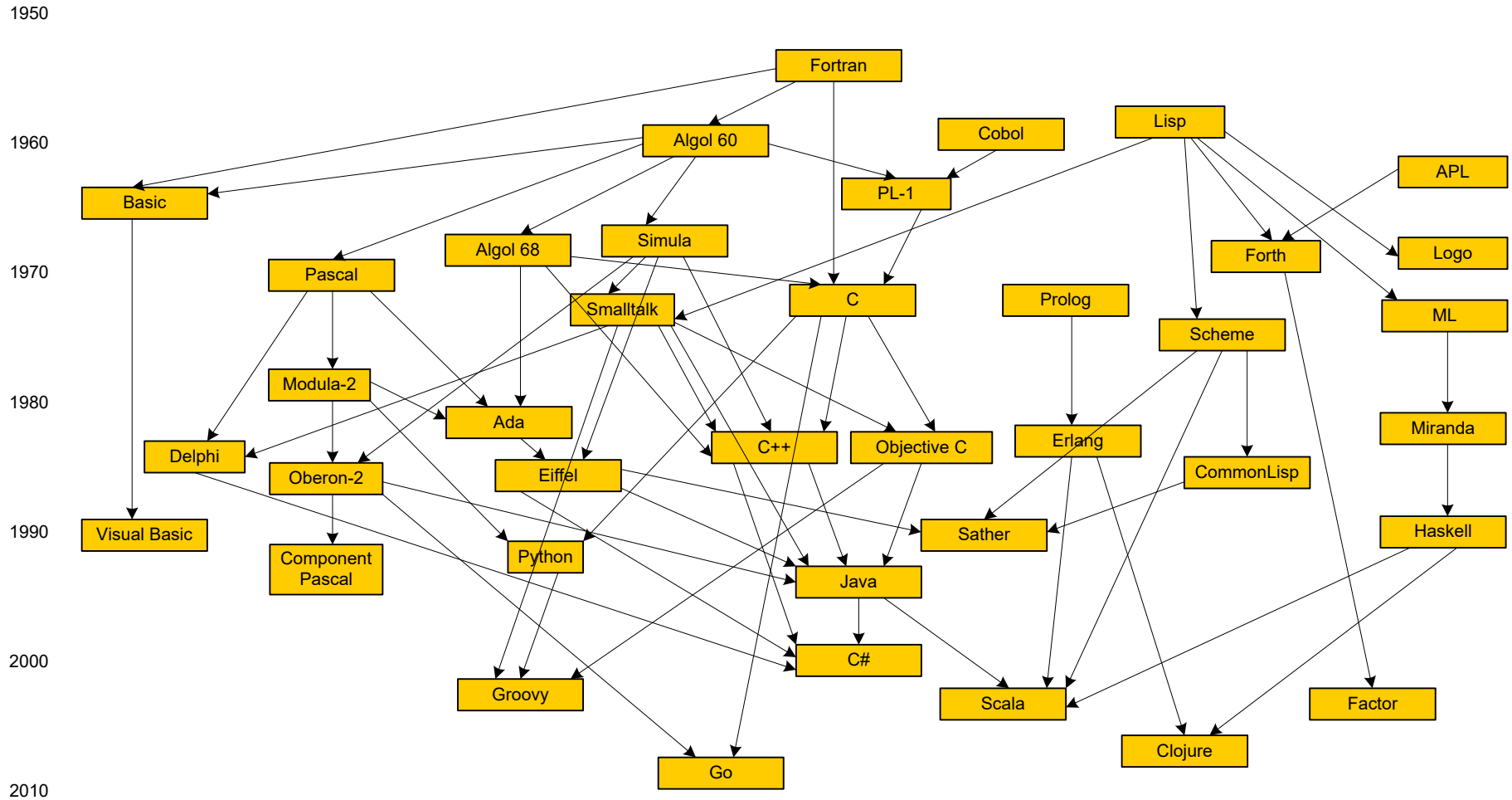
Zuse, Horst (1999): Geschichte der Programmiersprachen. Berlin: Technische Universität,

Höhere Programmiersprachen[1]

Jahr	Name	Abkürzung für	Bemerkungen
1957	Fortran	formula translation	Mutter aller algorithmischen Sprachen, binär
1960	Algol-60	algorithmic language	Ursprung der Algol-Familie mit bahnbrechenden Ideen
1960	Cobol	Common business oriented language	Für kommerzielle Anwendungen auch heute noch die am meisten verwendete Sprache. Für technische Anwendungen ungeeignet.
1962	Basic	Beginners all purpose symbolic instruction code	Einfache Sprache für Anfänger
1962	APL	A programming language	Mit Vektoren und Matrizen arbeitende Sprache mit speziellem Zeichensatz und vielen ungewöhnlichen Operatoren
1962	Lisp	List processing language	Hauptsprache der Künstlichen Intelligenz. Einzige Datenstruktur ist der binäre Baum (=Liste)
1965	PL/I	Programming language I	Sehr umfangreiche Sprache für technisch-wissenschaftliche und kommerzielle Anwendungen
1968	Algol-68	Algorithmic language	Umfangreicher und komplizierter Nachfolger von Algol-60. Nur an einigen Universitäten benutzt
1971	Pascal	Nach Blaise Pascal	Erste moderne Informatik-Sprache, mit großem Einfluss auf die weitere Sprachentwicklung
1972	Prolog	Programming in logic	Modelliert das logische Schließen. Besonders für KI geeignet. Einzige Datenstruktur ist der binäre Baum.
1973	C		Vielbenutzte Sprache, eng mit dem Betriebssystem Unix verbunden
1980	Modula-2		Nachfolgerin von Pascal, auf Bedürfnisse modularer Programmierung zugeschnitten
1980	Ada	Nach Ada King, Countess of Lovelace	Sehr umfangreiche Sprache der Pascal- und PL/I-Nachfolge
1980	Smalltalk		Die Pioniersprache des objektorientierten Denkmodells
1982	C++		Objektorientierte Erweiterung von C
1988	Oberon-2	Nach dem Uranus-Mond Oberon	Objektorientierte Sprache
1995	Java	Eigentlich Object Application Kernel (OAK)	"Internetsprache", unabhängig von Plattform

[1] Nach: Rechenberg, Peter (2000): Was ist Informatik? München: Hanser, S. 124

Programmiersprachenbaum[2]



[2] Gruntz, Dominik (2011): Einführung in Java. Vorlesungsfolie. Windisch: Hochschule für Technik FHNW