Deutschlandfunk Kultur, Zeitfragen-Feature vom 17.10.2018, 19.30 – 20 Uhr

"Auch Code hat eine Vergangenheit. Warum wir historisches Wissen über Programmiersprachen brauchen"

Von Florian Felix Weyh

---

001 Vocoder

run: history of programming languages

002 Leitenberger 0‘04

Mein erster Computer war ein TI-99, Schrägstrich 4a.

003 Schneider 0‘10

Der Rechner damals, 1959, der kostete ne halbe Million Mark, und das war eine Zuse Z-22.

004 Kappes 0’17

Ich hab in einem Urlaub – das weiß ich noch wie heute, weil, ich hatte gar keinen eigenen Computer – mir ein Buch mitgenommen, wie man den 6502-Prozessor versteht. Das war aber einfach nur 'ne Art Beschreibung des Prozessors. Also wie ein großes Datenblatt, auf 100 Seiten alle seine Befehle.

005 Leitenberger 0‘02

Ein Programm ist ja im Prinzip geistige Arbeit!

006 Kappes 0’10

Und dann hab ich im Sand – im Urlaub am Strand –, im Sand die Zustände der Register gemacht, indem ich da eben kleine Mulden in den Sand gedrückt habe und da Steinchen reingelegt habe.

007 Leitenberger 0‘16

Manche Leute schreiben gern Gedichte, andere Leute malen gerne, und da wollen sie auch ein schönes Gedicht haben oder ein schönes Bild! Und man kann ein Programm machen, dass es schnell hingerotzt ist. Oder man kann ein Programm machen, das schön ist. Also schön zum Beispiel, dass man es nach einem Monat noch verstehen kann, wenn man’s anguckt.

008 Vocoder

insert: basic question

01 AUTOR

Brauchen wir ein historisches Gedächtnis für Programmiersprachen?

009 Vocoder

define: history

010 Schneider 0‘10

Golo Mann hat einmal geschrieben: "Vergangenheit und Zukunft gehören begrifflich zusammen. Wer keine Vergangenheit kennt, wird keine Zukunft in den Griff bekommen."

011 Vocoder

name: source

02 AUTOR

Wo hat Golo Mann das geschrieben?

012 Schneider 0‘10

Das genaue Zitat weiß ich nicht, aber es ist so in der Süddeutschen Zeitung im Feuilleton 1971 gewesen. Soviel weiß ich noch. Das gilt ganz genauso für die Programmiersprachen! Man kann Java, was heute ja eine der aktuellen Sprachen ist – man kann’s benutzen! Aber wenn man verstehen will, warum das so jetzt aussieht, dann muss man eigentlich wissen was davor war.

013 David Lettermann (historischer O-Ton)

Please welcome Admiral Grace Hopper!

014 Vocoder (über den Applaus)

delete "Hopper", go to speaker

03 AUTOR

Professor Hans Jürgen Schneider, emeritierter Informatiker an der Friedrich-Alexander-Universität Erlangen-Nürnberg, hält seit dreißig Jahren jedes Semester eine Vorlesung zur Geschichte der Programmiersprachen.

015 Schneider 0‘13

Interessant ist, dass in der ersten Woche – oder vielleicht auch noch in der zweiten – eine größere Zahl Studenten kommt, so um die 30, 35. Und nach drei Wochen bleiben dann fünf oder sechs übrig.

016 Vocoder

false: students are interested

true: students are unmotivated

017 Jochen Ziegenbalg 0‘18

In welcher Weise sollten Programmiersprachen – oder sollte Programmierung – unterrichtet werden? Ich meine, es sollte immer so unterrichtet werden, dass die Dinge, die wahnsinnig schnell veralten nicht im Vordergrund stehen, sondern dass das, was eine längere Lebenszeit hat, dass das ist, was im Vordergrund steht!

04 AUTOR

Professor Jochen Ziegenbalg schrieb 1996 zusammen mit seinen Söhnen die Monografie "Algorithmen von Hammurapi bis Gödel". Mehrfach aktualisiert, liegt sie inzwischen in der vierten Auflage vor. Algorithmenkunde ist Einordnungswissen.

019 Kappes 0’20

Ich unterscheide immer – sage ich mal – Handhabungs- und Anwendungswissen von Einordnungswissen, und dann tatsächlich einer eher politisch-philosophischen Dimension. Politisch-Philosophisch ist das schon hilfreich zu wissen oder sogar nötig zu wissen, wie der Rechner im Detail funktioniert, weil erst dadurch verstehe ich, was eigentlich Determinismus ist, beispielsweise.

05 AUTOR

Christoph Kappes studierte Jura und Informatik, war Programmierer und gründete mehrere Firmen mit. Heute arbeitet er als Unternehmensberater und Publizist.

021 Kappes 0’20

Ich traue der Maschine eigentlich auch nicht so viel zu, weil ich sie noch gelernt hab als eine, die nur Nullen und Einsen hin- und herschiebt, und ich das auch verstehe. Also das hat nichts Mystisches. Das hat nichts Geheimnisvolles, aber auch nicht, was mich gefährdet. Also es macht mir auch keine Angst.

Musik

06 AUTOR

Angst ist ein schlechter Ratgeber, Code muss keine Angst machen, Code ist wenig erschreckend, wenn man ihn versteht. Kann man ihn verstehen? Natürlich, sonst gäbe es keine Programmierer. Kann ihn jeder verstehen? Da scheiden sich die Geister. Doch um die Auswirkungen von Code auf die Gesellschaft zumindest grob einschätzen zu können, wäre vielleicht historisches Wissen hilfreich, das sich mit der Entwicklung jener seltsamen Sprachen befasst, die niemand spricht.

01 ZITATOR WEIBLICH

"Eine Programmiersprache ist in erster Linie eine künstliche Sprache zur Kommunikation von Menschen mit Maschinen. Im Unterschied zu natürlichen Sprachen, die der Kommunikation zwischen Menschen dienen, sind Programmiersprachen vollkommen eindeutig definiert: Es gibt keinerlei Interpretationsspielraum, was ein bestimmtes Sprachkonstrukt bedeutet."[1]

022 Vocoder

name: source

02 ZITATOR WEIBLICH

Johannes Jander und Kathrin Passig: "Weniger schlecht programmieren". O'Reilly Verlag 2013.

023 Kappes 0‘23

Es sind sicherlich Sprachen in dem Sinne, dass da Zeichen eine Bedeutung haben, die man wissen muss. Aber die Eigenschaft von menschlicher Sprache ist ganz anders als die von Programmiersprachen. Weil bei der Programmiersprache das einmal definiert wird, was der Befehlssatz ... wie der heißt und was der bedeutet. Ab da ist die Sprache starr.

01 ZITATOR MÄNNLICH

"Streng genommen kann jeder Computer nur eine einzige Programmiersprache verarbeiten: seine Maschinensprache. Diese Maschinensprachen sind außerordentlich unanschaulich und schwer verständlich. Ein Maschinenprogramm besteht im Grunde nur aus einer einzigen langen Folge von 0/1-Symbolen. (…) Einen allerersten Schritt von den 0/1-Folgen der Maschinensprache in die Richtung der höheren Programmiersprachen stellten die sogenannten Assemblersprachen dar, bei denen die durch 0/1-Folgen ausgedrückten Maschinenbefehle wenigstens durch anschauliche, symbolische Namen beschrieben werden.[2]

024 Vocoder

name: source

02 ZITATOR MÄNNLICH

Jochen Ziegenbalg et altera: "Algorithmen von Hammurapi bis Gödel". Springer Verlag 2016.

025 Vocoder langsames Fadeout gegen Ende

list programming languages: Fortran, Cobol, Algol, Lisp, Basic, Simula, Pearl, Pascal, Prolog, Scheme, Ada, C, C++, Haskell, Python, SQL, Java, Ruby, Swift

026 Historischer O-Ton 0’37 über Vocoder

Fortran represents the most advanced coding systems available today.

027 Schneider über Vocoder 0‘12

Es gab mal vor vielen Jahren im Internet, als ich das angefangen habe mit der Vorlesung, eine Internetseite, auf der waren 8.000 Sprachen! Aber die ist nicht mehr weitergepflegt worden, und inzwischen ist sie verschwunden.

03 ZITATOR WEIBLICH

"Es gibt, je nach Zählweise, zwischen 500 und über 8.000 Programmiersprachen."[3]

07 AUTOR

… sagen auch Johannes Jander und Kathrin Passig, vermutlich in Bezug auf die gleiche Quelle. Davon dürften die meisten allerdings Abspaltungen und Dialekte sein, so dass sich die tatsächlich eigenständigen Programmiersprachen im niedrigen dreistelligen Bereich bewegen. Zwischen Programmiersprachen, der Tür zur Maschine, und Anwenderprogrammen, dem Fenster zur Welt, sitzt noch das Betriebssystem. Jedes Mal beim Booten des Computers macht es sich zeitraubend bemerkbar. Es regelt das Zusammenspiel aller technischen Komponenten und ist seinerseits in einer Programmiersprache geschrieben, heute meist C++. Ohne Betriebssystem geht nichts.

028 Schneider 0‘11

Selbst diese alte Zuse-22, mit der ich angefangen habe zu arbeiten, die hatte so etwas wie ein rudimentäres Betriebssystem. Das waren so ungefähr 2.000 Speicherplätze mit Befehlen.

029 WH David Lettermann (historischer O-Ton)

Please welcome Admiral Grace Hopper!

030 Vocoder

go to: year 1943

start: history of computing machines

031 Grace Hopper (historischer O-Ton) 0’11

I think we’ve totally forgotten the environment in which MARK I appeared. 41 was Pearl Harbour. By 43 we were in the thicker things.

08 AUTOR

Die Geschichte des Computers ist ohne eine überragende Frau undenkbar. Die an Neujahr 1992 verstorbene Grace Hopper – Admiral Grace Hopper – wurde 1906 geboren, promovierte in Mathematik und unterrichtete dieses Fach bis 1944 am Vassar College. Im Auftrag der amerikanischen Marine wurde sie dann nach Harvard zu einer Maschine abkommandiert, die ballistische Berechnungen durchführen sollte: die MARK 1.

03 ZITATOR MÄNNLICH

»Da stand diese wuchtige Riesenmaschine und machte eine Menge Lärm«, erinnert sie sich. »Es war alles offen und unverdeckt und sehr, sehr laut.« Ihr war klar, dass sie sie ganz und gar würde verstehen müssen, um damit richtig arbeiten zu können, und so verbrachte sie ihre Nächte über den Schaltplänen."[4]

032 Vocoder

name: source

04 ZITATOR MÄNNLICH

Walter Isaacson: "The Innovaters". Bertelsmann Verlag 2018.

9 AUTOR

Das Ungetüm MARK 1 war ein elektromechanisches Gerät, kein elektronisches, anders als der zeitgleich von den US-Landstreitkräften entwickelte Röhrenrechner ENIAC. Alle Schaltzustände wurden mit klackernden Relais erzeugt, deren Geschwindigkeitsnachteil gegenüber dem ENIAC sich schnell offenbarte.

05 ZITATOR MÄNNLICH

»Als man endlich ein bisschen über sie wusste«, so Hopper über die Mark I, »war sie schon komplett überholt, und alle hatten es nur noch mit Elektronik.«[5]

033 Leitenberger 0‘18

Der ENIAC, der hat ungefähr so 2.300 Röhren, und die waren zu bestimmten Funktionsgruppen angeordnet – der hat auch bloß Rechenaufgaben gelöst –, und wenn man jetzt irgendwo eine Addition vor einer Multiplikation haben wollten, dann musste man eben die Verkabelung der Funktionsgruppen neu machen, dass eben das auch so entsprechend lief.

034 Vocoder

Describe: speaker

12 AUTOR

Bernd Leitenberger lebt als Programmierer im schwäbischen Filderstadt. In seinem Buch "Computergeschichte(n)" zeichnet der Technikenthusiast den gewaltigen Sprung der Rechnerentwicklung seit 1944 anschaulich nach.

035 Leitenberger 0‘18

Es gab da mal einen Ausschnitt in der US-Wochenschau, als der schon ein Jahr lief. Er war ja auch geheim, war ein militärischer Computer. Da kam also wortwörtlich das Zitat: "Der ENIAC berechnet die Bahn eines Geschosses, das 30 Sekunden braucht, in 2 Sekunden! Das Programmieren dauerte 2 Tage."

036 Schneider 0‘09

Programmieren hieß damals: Ich stecke ein Kabel von der Einheit zu dieser Einheit. Und wenn die Einheit eine Sache ausgerechnet hat, schickt sie das über das Kabel an diese Einheit.

037 Konrad Zuse (historischer O-Ton, evtl. etwas verrauschen) 0‘16

So kann für den Rechenmaschinenbauer die Frage lauten: "Brauchen wir verschiedene Zwecke wie zum Beispiel die Lösung von linearen Gleichungen, die Lohnberechnung für eine Fabrik, die Steuerung einer Werkzeugmaschine verschiedene Geräte?

038 Vocoder

explain situation

13 AUTOR

Der deutsche Computerpionier Konrad Zuse, der hier in einem Radiovortrag von 1963 die Frage nach der universellen Einsetzbarkeit von Computern aufwirft, hatte ebenfalls in den 40er-Jahren einen elektromechanischen Rechner konstruiert. Ihn beschäftigte von Anbeginn die Frage nach der Hierarchie zwischen Geist und Materie. Wer beherrscht wen?

039 Konrad Zuse (historischer O-Ton, evtl. etwas verrauschen) 0‘17

Die Amerikaner haben hierfür die Ausdrücke "Hardware" und "Software" eingeführt. Das heißt, man kann ein Problem schaltungsmäßig mit geschickter Verknüpfung von konstruktiven, also gewissermaßen harten Elementen oder auf dem Wege der Programmierung lösen.

14 AUTOR

Lange bevor der Begriff "Software" überhaupt aufkam, hatte Zuse schon seine erste Programmiersprache entworfen: den "Plankalkül". Allerdings nur auf Papier, als reine Idee.

06 ZITATOR MÄNNLICH

"Das Manuskript wurde von Konrad Zuse von 1942-1945/46 erstellt, aber der internationalen Öffentlichkeit erst 1972 bekannt. Der Plankalkül war aus heutiger Sicht eine bemerkenswert vollständige Programmiersprache (…) [und] wurde niemals implementiert."[6]

15 AUTOR

… berichtet Zuses Sohn Horst, selbst auch Informatiker.

040 Vocoder

name: source

07 ZITATOR MÄNNLICH

"Geschichte der Programmiersprachen" von Horst Zuse. Technische Universität Berlin, 1999.

041 Vocoder

go to: "Logic Theorist"

16 AUTOR

Wer beherrscht wen? Auf den ersten Blick sieht es einfach aus: Hardware gibt die Möglichkeiten der Programmiersprachen vor, die vom Menschen erdachte Syntax begrenzt die Kombinationsmöglichkeiten logischer Prozeduren – zumindest in den

sogenannten "imperativen Sprachen" der frühen Jahre. Doch schon 1956 strebte der spätere Wirtschaftsnobelpreisträger Herbert A. Simon zusammen mit dem Kognitionspsychologen Allen Newell das an, was wir heute "Künstliche Intelligenz" nennen. Beide skizzierten auf wenigen Schreibmaschinenseiten ein Programm, das ohne imperative Vorgaben im Detail selbstständig 38 Theoreme aus Bertrand Russells Principia Mathematica lösen sollte. Sie nannten es "Logic Theorist".

08 ZITATOR MÄNNLICH

"Der Beweis des Satzes 2.85 war tatsächlich eleganter als der Beweis, den Russell und Whitehead mühsam von Hand erstellt hatten. Simon war in der Lage, den neuen Beweis Russell selbst zu zeigen, der ‚mit Freude darauf reagierte'. Sie versuchten, den neuen Beweis im Journal of Symbolic Logic zu veröffentlichen, aber er wurde mit der Begründung zurückgewiesen, dass ein neuer Beweis eines elementaren mathematischen Theorems nicht bemerkenswert wäre – dabei übersehend, dass einer der Autoren ein Computerprogramm war."[7]

17 AUTOR

Im Gegensatz zu Zuses "Plankalkül" war der "Logic Theorist" kein papiernes Typoscript geblieben, sondern vom Programmierer Cliff Shaw mittels eigener Sprache namens IPL auf einem RAND-Computer implementiert worden. Die Absicht, dem Computer Denkautonomie beizubringen, statt ihm menschlich geprägte Logik aufzuzwingen, bestand also schon früh, wird aber erst heutzutage mit der Entwicklung künstlicher neuronaler Netzwerke realistisch. Zunächst hieß es kleine Brötchen backen.

042 Vocoder

jump back to "MARK I"

recapitulate female gender

043 Grace Hopper (historischer O-Ton) 0’12

In the first place she was the first large scale digital computer in the United States. She was automatically sequenzed. She was programmed. Step by step programming, exact as we have today.

18 AUTOR

Die Mark I ließ sich wie heutige Rechner binär programmieren, und "Amazing Grace", wie Grace Hopper von ihren Bewunderern genannt wurde, wechselte endgültig von der Mathematik zur Informatik und nahm dabei in die neue Disziplin jenes Gebot der Verständlichkeit hinüber, das schon ihre Arbeit an der Universität geprägt hatte:

09 ZITATOR MÄNNLICH

"Ihren Statistikkurs begann sie mit einer Vorlesung über ihre Lieblingsformel und ließ die Studenten anschließend einen Aufsatz darüber schreiben, den sie nach Klarheit des Ausdrucks und Schreibstil benotete. »Ich (…) erntete erboste Proteste, man habe schließlich einen Mathekurs belegt und keinen Englischkurs«, erinnert sie sich. »Dann erklärte ich, dass es sinnlos sei, Mathematik zu lernen, wenn man sein Wissen nicht mit anderen Menschen zu teilen vermöchte."[8]

Musik: Tom Lehrer "That’s Mathematics" Liedzeile: "When a ball / Bounces off of a wall / When you cook / From a recipe book / When you know / How much money you owe / That's mathematics!"

19 AUTOR

Konsequenterweise bemühte sich Hopper, die Programmierung von Computern in eine leichter als die Maschinensprache verstehbare Form zu überführen und entwickelte mit dem A-0-System den ersten funktionierenden Compiler der Welt.

044 Ziegenbalg 0‘18

Wenn man ein bestimmtes Programm vor sich sieht – vielleicht aus 10 Zeilen oder aus 100 Zeilen oder wie auch immer – dann gibt es eine Form der Übersetzung, das ist die sogenannte Compilierung. Die nimmt das Gesamtprogramm und übersetzt es auf einen Schlag in die Maschinensprache. Ausgeführt wird immer nur das Maschinenprogramm!

045 Atmo Christoph Kappes zählt mit den Fingern binär und verbiegt sie dazu unter merklicher Anstrengung. Vocoder darüber:

20 AUTOR

Christoph Kappes zählt binär, indem er seine Finger und Fingerzwischenräume als Null und Eins definiert. Das geht sogar. Aber nur sehr begrenzt. Schnell wäre der Mensch mit seinen Fingerfertigkeiten, vor allem aber mnemotechnisch überfordert. So erfand man immer mehr Schichten, die den eigentlichen Kern des Rechners umhüllen. Der Maschinencode wurde überformt von Programmiersprachen. Compiler, die ein Programm als Ganzes übertragen – oder Interpreter, die das zeilenweise erledigen –, übersetzen die Anweisungen zurück in Maschinencode.

046 Kappes 0’20

Die ganze Welt besteht aus Schichten, und auf der untersten Schicht ist halt ein Hardware-Layer, der bestimmte Transistoren hat und die irgendwie verdichtet auf so großen Siliziumflächen sind. Und man muss sich mit unteren Schichten nicht befassen, die sind im Grund nur gekoppelt, könnte man sagen. Also man muss nicht auf die untere Schicht gehen.

21 AUTOR

Zumindest nicht in der Anwendungsprogrammierung, die längst modular erfolgt. Dabei greift man auf Bibliotheken bereits vorhandenen Codes zurück, ohne im Einzelnen zu wissen, wie die gewünschte Funktion auf einer tieferen Ebene programmiert ist. Aber heißt Geschichte nicht auch Entschichtung, Schichtenfreilegung, Grabung nach der Urschicht?

047 Vocoder

recall: assembler

048 Krakenbürger 0‘14

Assembler ist einfach von der Logik her eine Sprache, mit der man anfangen kann, weil die wirklich so beim Urschleim anfängt und man nicht auf Anhieb produktiv ist und sofort ne Web-App entwickeln kann. Aber man versteht erstmal ziemlich viel darüber, wie ein Computer funktioniert.

049 Vocoder

describe: speaker

22 AUTOR

Die Berliner Soziologin Fiona Krakenbürger betrieb vor einigen Jahren den Blog "Fiona lernt programmieren". Darin dokumentierte sie vier Monate lang ihren – freilich von erfahrenen Freunden gestützten – Versuch, ohne Vorkenntnisse eine Ur-Programmiersprache zu lernen: den eingangs erwähnten Assemblercode, erfunden im Holozän des Computerzeitalters zwischen 1948 und 50.

050 Krakenbürger 0‘06

Wir haben so dies und das gemacht, immer mit dem Ziel, Assembler zu verstehen.

23 AUTOR

Bewusst oder unbewusst trat sie damit in die Fußstapfen des 2011 verstorbenen Literaturwissenschaftlers Friedrich Kittler, der sich schon frühzeitig mit der fremden technischen Sprache beschäftigt und sogar die Assemblerprogrammierung erlernt hatte. Sich Zugangswissen zur Maschine anzueignen, hielt er für eine intellektuelle Pflicht. 1991 warnte er in einem Aufsatz vor der Tendenz der damals noch keineswegs komfortablen PC-Technologie, sich von den maschinennahen Bedienungsroutinen zu entfernen:

09 ZITATOR MÄNNLICH

"Unter Stichwörtern wie Benutzeroberfläche, Anwenderfreundlichkeit oder auch Datenschutz hat die Industrie den Menschen mittlerweile dazu verdammt, Mensch zu bleiben. (...) Je höher und komfortabler die Hochsprachen, desto unüberbrückbarer ihr Abstand zu einer Hardware, die nach wie vor alle Arbeit tut."[9]

051 Vocoder

name: source

10 ZITATOR MÄNNLICH

Friedrich Kittler: "Protected Mode". In "Draculas Vermächtnis". Reclam Verlag.

24 AUTOR

Auf dieser Einschätzung, dass das Wissen um Programmiersprachen zur emanzipativen Grundausstattung des Menschen gehören sollte, beruhte auch Fiona Krakenbürgers Selbstversuch.

052 Krakenbürger 0‘12

Manchmal ist vielleicht so ein komplett nutzloses Herangehen an Technologielernen der richtige Weg! Weil’s nicht darum geht: "Du musst programmieren lernen und dann Entwicklerin werden!" Sondern: "Ne, schau’s dir doch einfach mal an! Guck mal, wie das funktioniert!"

25 AUTOR

… und zieh daraus dann deine Schlüsse, was im Falle der damaligen Ethnologiestudentin überraschend deutlich ausfiel. Krakenbürger wechselte zur Techniksoziologie und arbeitet heute in einem Open-Knowledge-Projekt, das die Kittlerschen Bedenken aus den 90er-Jahren einer damals erst heraufziehenden, heute längst manifesten IT-Untertanenmentalität indirekt aufgreift. Je perfekter die Geräte, desto verborgener das Wissen um ihr Funktionieren. Es wird, schrieb Kittler 1991 ...

11 ZITATOR MÄNNLICH

"... auf der (...) benutzerfreundlich kaschierten Seite nachgerade unmöglich, vom Fertigprodukt auf seine Produktionsbedingungen zurückzuschließen oder diese Bedingungen gar zu verändern."[10]

26 AUTOR

Das allerdings ist ein alter, gegen die Interessen der Computerindustrie gerichteter Traum der Hacker-Community: Unter der lackierten Oberfläche von elektronischen Konsumgeräten lauern mehr Möglichkeiten, als die Bedienungsanleitungen preisgeben.

053 Krakenbürger 0‘27

Ein gutes Beispiel dafür ist ein Tamagotchi! Es gibt einen tollen Talk von einer Hackerin, die sehr, sehr viel Zeit damit verbracht hat, ihren Tamagotchi zu reverse-engeneeren. Die hat den aufgemacht und dann versucht, den Programmcode rauszukratzen. Und das lag eben in Assemblercode vor – also in einer Art Assemblercode – und damit hat sie dann versucht rauszufinden: "Okay, wie ist dieser Tamagotchi programmiert", und hat dann so nach und nach sich rangetastet, den Code entschlüsselt, und kann jetzt ihren Tamagotchi anpassen an ihre eigenen Bedürfnisse!

27 AUTOR

Sich Assembler anzueignen, ist also eine Möglichkeit, Herrschaftsformen zu brechen – ganz wie es sich Friedrich Kittler erhoffte. Beim Tamagotchi, diesem elektronischen Kleinspielzeug, erscheint das noch niedlich. Wird das Herrschaftswissen allerdings vom Staat benutzt, um seine eigenen Regeln zu brechen, bekommt die Hacker-Anarchie einen Anstrich von Notwendigkeit.

054 Vocoder

go to: October 9th, 2011

055 Padeluun (historischer Telefon-O-Ton) 0‘16

Also diese wunderbaren Feuilletonseiten der Frankfurter Allgemeinen Sonntags-zeitung sind ein Dokument der Zeitgeschichte! Wer noch ein Exemplar kriegen kann, sollte sich das holen und sich das an die Wand hängen und ganz groß drüberschreiben: "Code breakes law!" Also ein Programmcode bricht Gesetze!

12 ZITATOR MÄNNLICH

"Eine Schadsoftware zu analysieren ist vergleichbar mit der Obduktion einer unbekannten Spezies von Lebewesen. (…) Aus dem Vorhandensein oder auch Fehlen bekannter Strukturen erschließt man wahrscheinliche Funktionen und Zusammenhänge der Anatomie. Mit vergleichbaren Methoden identifiziert man auch Funktionen und Zusammenhänge in einer unbekannten Schadsoftware, die ja nur als Maschinencode vorliegt."[11]

056 Vocoder

name: source

13 ZITATOR MÄNNLICH

Frank Rieger in der "Frankfurter Allgemeinen Sonntagszeitung" vom 9. Oktober 2011.

28 AUTOR

Die Schadsoftware, deren Assemblercode die FAS zeitungsseitenbreit abdruckte, war von den Hackern des Chaos Computer Clubs aus dem sogenannten "Bundestrojaner" extrahiert worden – einer versteckten und im ermittelten Funktionsumfang illegalen staatlichen Spionagesoftware. Nach einigem Tumult im politischen Raum wurde sie stillschweigend durch ein anderes Überwachungstool ersetzt, dessen Programmzeilen bis heute unbekannt geblieben sind. Normalsterbliche würden sie allerdings ebenso wenig verstehen wie assyrische Keilschrift, ägyptische Hieroglyphen oder die Zeitungsseite 2011 voller Assemblercode. All dies erscheint Laien wie ein grafisches Flächenmuster, das allenfalls implizit einen Appell enthält:

057 Vocoder

repeat: history

058 WH Schneider 0‘1

"Wer keine Vergangenheit kennt, wird keine Zukunft in den Griff bekommen."

29 AUTOR

Und zumindest was den immer noch hoch relevanten Assemblercode angeht, ist der Golo-Mann-Satz übertragbar auf die Technikgeschichte.

Musik: "Master of Magic", Mittelteil-Akkord

059 Leitenberger 0‘16

Wenn man so ne Hierarchie mal baut: Assembler, sagt man meistens, ist die Programmiersprache der ersten Generation. Fortran und Cobol als spezialisierte

Programmiersprachen für einen bestimmten Verwendungszweck sind die zweite Generation. Und was danach kommt – das sind die meisten Programmiersprachen, die es heute gibt – ist die dritte Generation.

060 WH David Lettermann (historischer O-Ton)

Please welcome Admiral Grace Hopper!

20 AUTOR

"Amazing Grace" taucht noch einmal maßgeblich in der Geschichte der Programmiersprachen auf. Sie entwickelte Ende der 1950er-Jahre die "Common Business Oriented Language" mit – kurz Cobol –, die bis heute gleichsam als Kobold in den Tiefen unseres Finanzsystems herumspukt. Immer noch.

061 Leitenberger 0‘23

Damit kann man sehr gut Daten verarbeiten! Also zum Beispiel wird bei Cobol definiert, wie Daten formatiert werden, damit kann man dann relativ gut Konten führen. Man kann mit dem aber fast gar nicht rechnen, weil zum Beispiel selbst Rechenoperationen als Worte geschrieben werden! Also anstatt "1+2" schreiben Sie: "1 ad 2". Oder "five multiply by three".

31 AUTOR

Das passt zur Mathematikerin, die einst bei ihren Studenten Stil und Ausdruck benotete, aber nicht dafür wurde die Sprache berühmt-berüchtigt. Sie steht für die bislang größte Hysterie des Computerzeitalters, den Y2K-Bug. Trotz aufwändiger Sprachbefehle legte man bei Cobol nämlich Wert auf Datensparsamkeit, weswegen für Jahrenzahlen nur zwei Ziffern bereitstanden. Diese Konvention übernahmen zahllose Anwenderprogrammierer.

062 Ziegenbalg 0‘18

Ich kann mich an eine Situation erinnern, wo ich mal als Werkstudent für ne Firma Programme geschrieben habe, und ich habe schon damals – das war Ende der 70er-Jahre – hab ich dieser Firma gesagt: "Ihr werdet im Jahr 2000 in ein Problem laufen!" Und das ist prompt passiert, nicht nur bei dieser Firma, sondern auch bei anderen Firmen.

32 AUTOR

Der Systemfehler, durch den die Menschheit an Silvester 1999 den elektronisch verursachten Weltuntergang erwartete, ließ sich allerdings im Vorfeld beheben. Interessant bleibt dennoch die Frage, warum die Zukunft bei Software-Entwicklern eine so geringe Rolle spielt?

063 Vocoder

jump to future

064 Bernd Leitenberger 0‘13

Sie können ja nicht irgendwo ein System entwerfen, das für die Zukunft ist, weil Sie erstens nicht wissen, wie die Zukunft wird. Und zweitens ein zweitens ein solches System, das auf Zukunft entworfen ist, wird viel komplizierter sein als das, was Sie derzeit brauchen. Das bezahlt Ihnen ja auch keiner.

065 Ziegenbalg 0`11

Diese alten Programmcodes haben ne lange Lebenszeit schon hinter sich, und mit dieser langen Lebenszeit signalisieren sie ihren Benutzern auch ne gewisse Verlässlichkeit!

04 ZITATOR WEIBLICH

"Der Wert eines alten Systems ist axiomatisch. Je länger das System funktioniert hat, desto größer die Zahl der Programmierer, die daran gearbeitet haben, und um so weniger versteht es ein einzelner Mensch. (...) Es läuft. Das ist seine Daseinsberechtigung, es leistet nützliche Arbeit. Egal wie schlecht, egal wie fehlerhaft, egal wie überflüssig – es läuft."[12]

066 Vocoder

name: source

05 ZITATOR WEIBLICH

Ellen Ullman, "Close to the machine", Suhrkamp Verlag 1999.

33 AUTOR

Obwohl zwei Dekaden vergangen sind, seit die amerikanische Programmiererin ihren bis heute lesenswerten, kulturkritischen Einblick in die Welt der Software-Entwicklung veröffentlichte, hat sich am Kernproblem wenig geändert.

06 ZITATOR WEIBLICH

"Wir wollen das Neue und am liebsten das Allerneueste. Nur Software darf altern, denn zuviel Zeit wird in sie investiert, und man braucht zuviel Zeit, um sie zu ersetzen. Also bastelt man, anders als bei aussortierter Hardware, an Software herum. Sie wird gewartet und repariert, geflickt und wiederverwendet."[13]

067 Bernd Leitenberger 0`25

Ich bin ja auch Softwareentwickler. Das heißt, ich muss mich dann mit Kunden rumschlagen. Dann sage ich auch des Öfteren: "Da können wir aber doch ein bisschen weiterdenken, Ihr wollt ja irgendwann mal das und das machen!" – "Ja, aber erstmal so!" Oder: "Erstmal einen schnellen Workaround!" Das ist ja meistens immer das ... ein schneller Workaround! Das ist ja meistens … heißt, wir haben gerade ein Problem oder irgendwas läuft jetzt anders, als es sein sollte, und dann tun wir eben das Problem mal umbiegen! Also wir gucken drauf, dass wir eigentlich eine schnelle, praktikable Lösung haben.

34 AUTOR

Was Bernd Leitenberger in der Praxis erlebt, weist wie Ullmans Notate auf ein Paradoxon hin: Software wird einerseits für den Tag und höchstens für ein kurzes Morgen geschrieben, überlebt dann andererseits aber Jahrzehnte kraft des Faktischen: Weil es sie gibt. Der intellektuelle Horizont der Informatiker ist stets nach vorne gerichtet … doch nach hinten plagen sie sich mit Patches, Improvisationen, Anpassungen, Schnittstellenmanagement und Datenportabilität ab. Weil häufig der "schnelle Workaround" Priorität hat, gehört Flickschusterei zum Alltag – oder wie es Johannes Jander und Kathrin Passig nennen: einen Kredit aufzunehmen.

07 ZITATOR WEIBLICH

"Bewusst schlechten Code zu schreiben, ist wie Schulden zu machen: Es ist sozial nicht gerne gesehen, aber in bestimmten Lebenslagen ist es dennoch die einzige Wahl. So, wie man Kredite später einmal – mit Zinsen – zurückzahlen muss, zahlt man eine Strafe dafür, wenn man Code hinschludert, um schnell an ein Ziel zu gelangen: Man hat später die undankbare Aufgabe, den wirren Codefluss zu entflechten."[14]

35 AUTOR

Und das schafft – wie alle Schulden – eine Rückbindung an die Vergangenheit. Man kann sie nicht hintergehen und nicht ungeschehen machen. Man kann sich aber zum eigenen Vorteil mit dem beschäftigen, was aus der Vergangenheit in die Gegenwart hinüberwächst.

068 Krakenbürger 0‘09

Ich wünsche mir sehr, dass man Leute auch dazu bringt sich zu trauen, das wirklich in seinen Grundzügen zu verstehen. Auch wenn man das vielleicht nie wieder braucht – aber es ist cool, es zu wissen!

Musik

070 Vocoder darüber

program successfully completed

now start from beginning

[1] Johannes Jander und Kathrin Passig: "Weniger schlecht programmieren". O'Reilly 2013. (zit. nach eBook)

[2] Jochen Ziegenbalg et.al.: "Algorithmen von Hammurapi bis Gödel". Springer Verlag 2016, S 223-224.

[3] Jander/Passig a.a.O.

[4] Walter Isaacson: "The Innovaters". Bertelsmann Verlag 2018. S. 118

[5] ebd. S. 123

[6] Horst Zuse "Geschichte der Programmiersprachen", Typoscript der TU Berlin, 1999. S. 6

[7] https://en.wikipedia.org/wiki/Logic_Theorist (Übersetzung vom Autor. Original: The proof of theorem 2.85 was actually more elegant than the proof produced laboriously by hand by Russell and Whitehead. Simon was able to show the new proof to Russell himself who "responded with delight". They attempted to publish the new proof in The Journal of Symbolic Logic but it was rejected on the grounds that a new proof of an elementary mathematical theorem was not notable, apparently overlooking the fact that one of the authors was a computer program.)

[8] Isaacson S. 117

[9] Friedrich A. Kittler: "Protected Mode" in: "Draculas Vermächtnis". Reclam Leipzig 1993. S. 209-210

[10] ebd.

[11] FAS vom 9.10.2011

[12] Ellen Ullman "Close to the machine", Suhrkamp Verlag 1999 [Original 1997]. S. 116

[13] ebd. S. 115

[14] Jander und Passig a.a.O.