

Projekte. Beratung. Spezialisten.



Die neue Kollegin ist da

Der Weg der Software-Entwicklung bis zur KI

IKS-Thementag 2024

Jan Radeck, Jörg Vollmer

19.06.2024

Agenda

› Von der Pascaline zur denkenden Maschine

- ◆ Hardware
- ◆ Software
- ◆ Entwicklungsprozess
- ◆ KI

› Die neue Kollegin ist da, und jetzt?

Timeline

1640

1840

1930

1940

1950

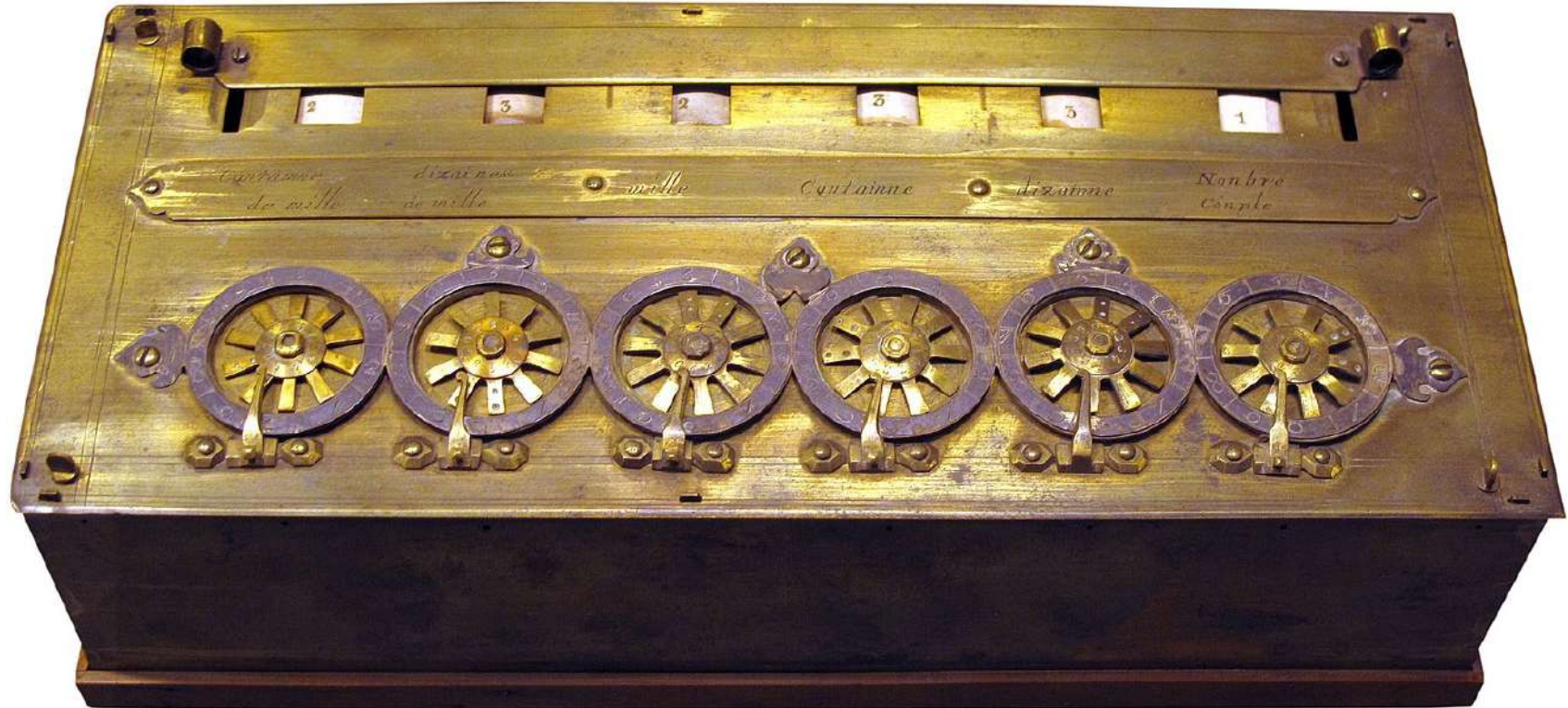
1960

1970

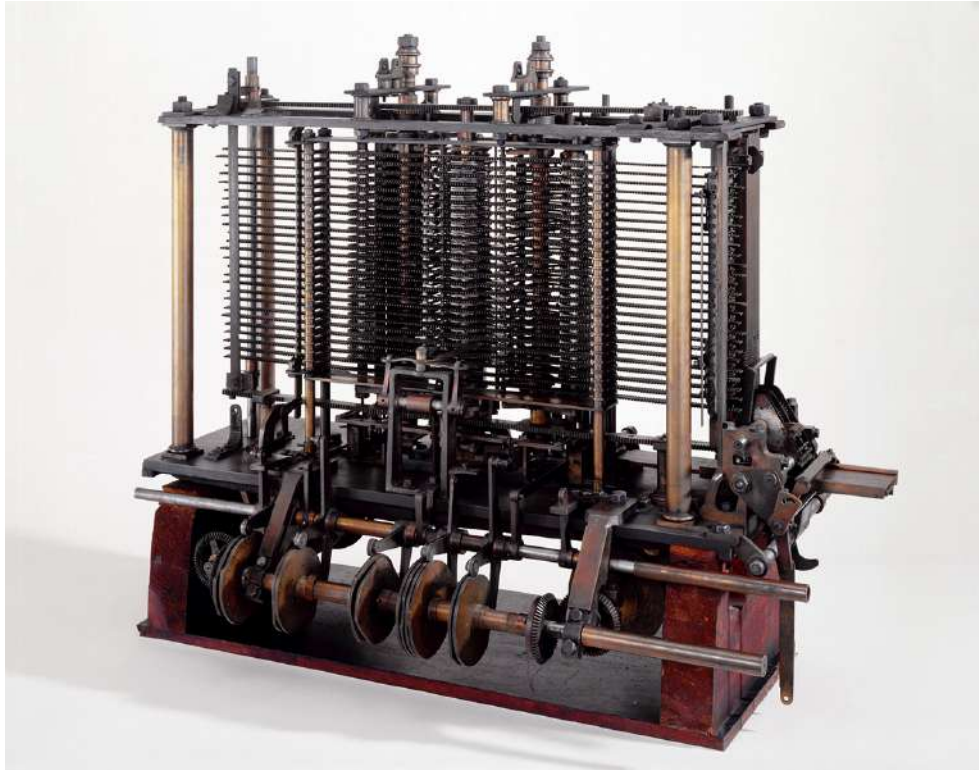
Hw

Hardware

1642: Die Pascaline, Blaise Pascal



1837: Analytical Engine, Charles Babbage



Die neue Kollegin ist da



1843: Die ersten Computer-Programme, Ada Lovelace

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Statement of Results.	Data.												Working Variables.				Result Variables.			
		V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}	V_{19}	V_{20}
		$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{8}$	$\frac{1}{9}$	$\frac{1}{10}$	$\frac{1}{11}$	$\frac{1}{12}$	$\frac{1}{13}$	$\frac{1}{14}$	$\frac{1}{15}$	$\frac{1}{16}$	$\frac{1}{17}$	$\frac{1}{18}$	$\frac{1}{19}$	$\frac{1}{20}$
1	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	$V_1 - V_2 = -1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	$V_1 + V_2 = 3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4	$V_1 \div V_2 = \frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
6	$V_1 - V_2 = -1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
7	$V_1 + V_2 = 3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
8	$V_1 \div V_2 = \frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
9	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10	$V_1 - V_2 = -1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
11	$V_1 + V_2 = 3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
12	$V_1 \div V_2 = \frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
13	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
14	$V_1 - V_2 = -1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
15	$V_1 + V_2 = 3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
16	$V_1 \div V_2 = \frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
17	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
18	$V_1 - V_2 = -1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
19	$V_1 + V_2 = 3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	$V_1 \div V_2 = \frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	$V_1 - V_2 = -1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
23	$V_1 + V_2 = 3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
24	$V_1 \div V_2 = \frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
25	$V_1 \times V_2 = V_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

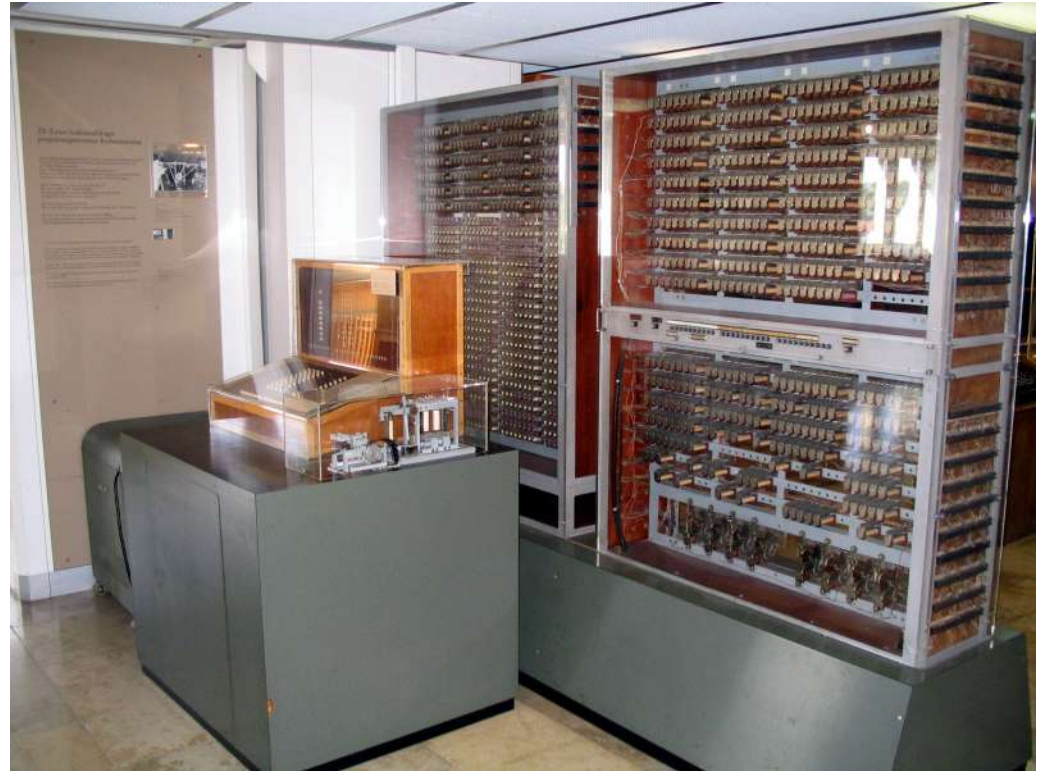
Here follows a repetition of Operations thirteen to twenty-three.



1930er: Das Relais, verwendet in der Z3



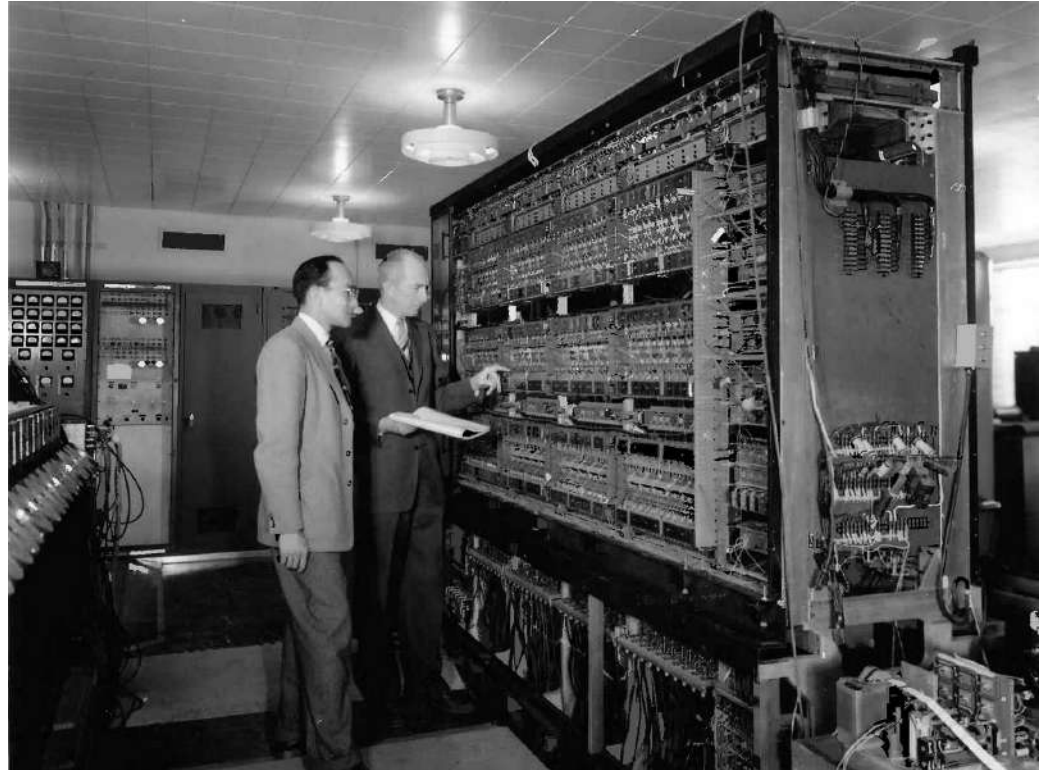
Die neue Kollegin ist da



1940er: Die Elektronenröhre, ENIAC, Universität von Pennsylvania



Die neue Kollegin ist da



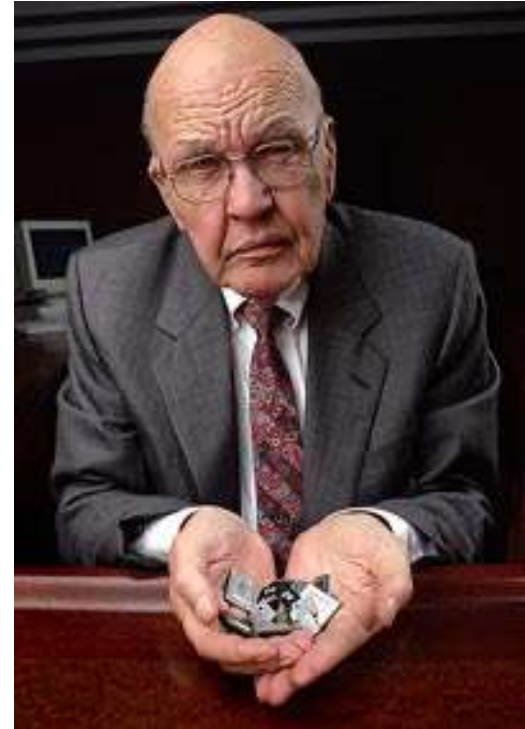
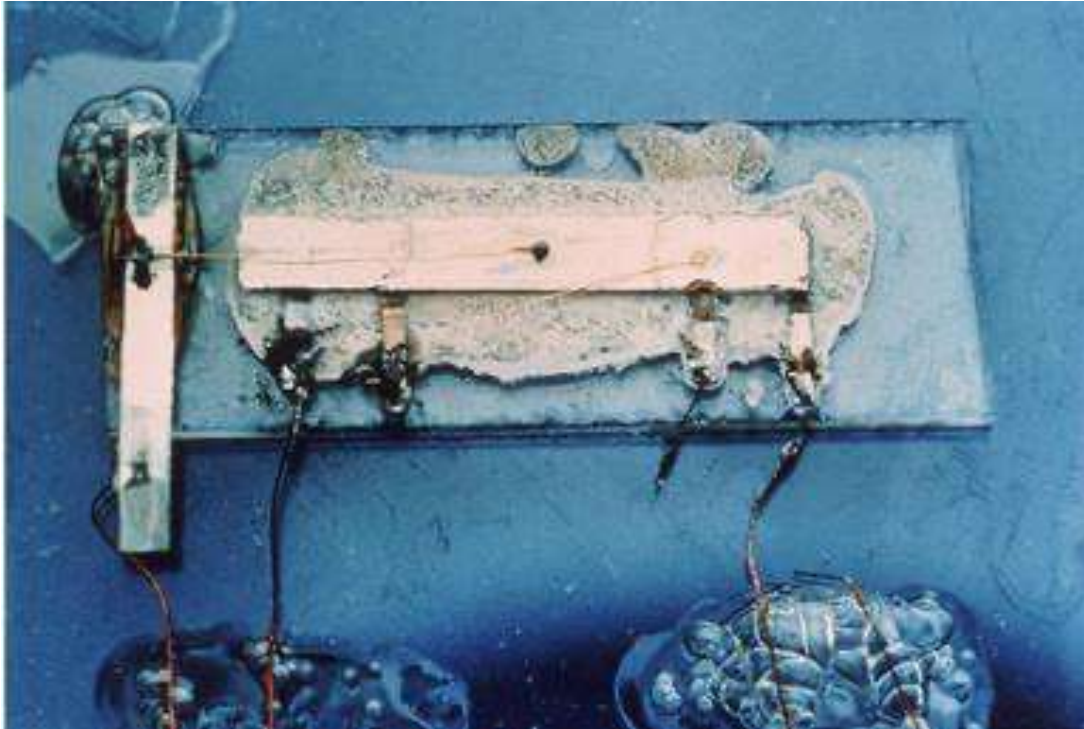
1950er: Der erste Transistor-Computer: TX-0, MIT



Die neue Kollegin ist da

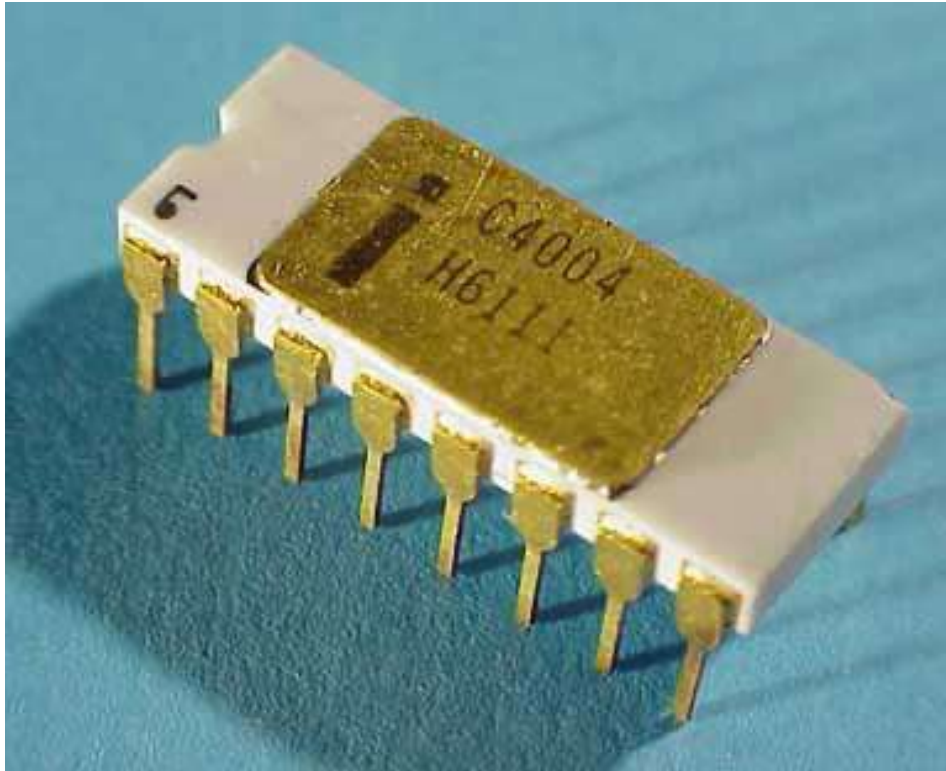


1960er: Der erste IC, Texas Instruments



Jack Kilby (Nobelpreis 2000)

1970er: Der erste Mikroprozessor, Intel 4004



Die neue Kollegin ist da

Timeline

1640

Pascaline



1840

Analytical Engine



1930

Zuse: Z3



1940

ENIAC



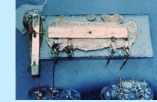
1950

TX-0



1960

IC



1970

Microprozessor



Hw

Sw

Software

1950er: Maschinen-Code

11100000 00000010	E0 02
11110000 00000010	F0 02
11100000 00000001	E0 01
11110000 00000010	F0 02
10100010 00000010	A2 02
11001010	CA
11110000 00000010	F0 02
11101100 00010000 00000000	EC 10 00
11110000 00000010	F0 02
11100000 00000001	E0 01
11110000 00000010	F0 02
10010000 11110000	90 F0
00111000	38
01100000	60
00011000	18
01100000	60

1960er: Assembler

```
start:
    LDA number
    JSR is_prime
    BCS not_prime
is_prime:
    CPX #2
    BEQ prime_done
    CPX #1
    BEQ prime_done
    LDX #2
check_divisible:
    DEX
    BEQ prime_done
    CPX number
    BEQ prime_done
    CPX #1
    BEQ prime_done
    BCC check_divisible
    SEC
    RTS
prime_done:
    CLC
    RTS
```

1960er: Assembler

```
start:
    LDA number           ; Lade die zu testende Zahl in das Akkumulator-Register
    JSR is_prime         ; Rufe die Unterfunktion auf, um zu prüfen, ob die Zahl eine Primzahl ist
    BCS not_prime        ; Wenn das Carry-Bit gesetzt ist, ist die Zahl keine Primzahl
is_prime:
    CPX #2               ; Vergleiche die Zahl im Akkumulator-Register mit 2
    BEQ prime_done       ; Wenn die Zahl gleich 2 ist, ist sie eine Primzahl
    CPX #1               ; Vergleiche die Zahl im Akkumulator-Register mit 1
    BEQ prime_done       ; Wenn die Zahl gleich 1 ist, ist sie keine Primzahl
    LDX #2               ; Setze X auf 2, um die Teilbarkeit durch alle Zahlen von 2 bis zur Hälfte der zu testenden Zahl zu überprüfen
check_divisible:
    DEX                 ; Dekrementiere X, um die nächste Zahl zu überprüfen
    BEQ prime_done       ; Wenn X gleich 0 ist, wurde keine Teilbarkeit gefunden, die Zahl ist eine Primzahl
    CPX number           ; Vergleiche die Zahl im Akkumulator-Register mit der aktuellen X-Zahl
    BEQ prime_done       ; Wenn die Zahl gleich der aktuellen X-Zahl ist, ist sie eine Primzahl (nicht teilbar)
    CPX #1               ; Vergleiche die Zahl im Akkumulator-Register mit 1
    BEQ prime_done       ; Wenn die Zahl gleich 1 ist, ist sie keine Primzahl
    BCC check_divisible ; Wenn die Zahl nicht teilbar ist, setze die Schleife fort
    SEC                 ; Setze das Carry-Bit, um anzuzeigen, dass die Zahl nicht prim ist
    RTS                 ; Rückkehr zur aufrufenden Stelle
prime_done:
    CLC                 ; Lösche das Carry-Bit, um anzuzeigen, dass die Zahl prim ist
    RTS                 ; Rückkehr zur aufrufenden Stelle
```

1960er: Fortran I

```
      READ(5,2) NUM
2     FORMAT(I)
      I = 2
3     IF(I**2-NUM) 4,5,5
4     PRINT 6
      STOP
5     IF(I-1) 6,7,7
6     I = I + 1
      GOTO 3
7     PRINT 8
      STOP
8     FORMAT(1H1, ' ', I5)
      END
```

1960er: Cobol

DATA DIVISION.

WORKING-STORAGE SECTION.

01 Num PIC 9(5).

01 IsPrime PIC X.

PROCEDURE DIVISION.

Accept-Num.

Perform Prime-Number-Test.

STOP RUN.

Accept-Num.

ACCEPT Num.

Prime-Number-Test.

IF Num <= 1

MOVE "N" TO IsPrime

ELSE

PERFORM Check-Divisors UNTIL Num
<= Sqrt(Num)

IF IsPrime IS INITIAL

MOVE "Y" TO IsPrime

END-IF

END-IF.

Check-Divisors.

DIVIDE Num BY Num - 1 GIVING
Remainder

IF Remainder = 0

MOVE "N" TO IsPrime

END-IF

SUBTRACT 1 FROM Num.

1960er: Algol 60

```
PROCEDURE is_prime_number(n);  
  INTEGER n, i;  
  is_prime := TRUE;  
  FOR i := 2 STEP 1 UNTIL sqrt(n) DO  
    IF n MOD i = 0 THEN  
      is_prime := FALSE;  
      EXIT;  
    FI;  
  OD;  
END;
```

1960er: Lisp

```
(defun is-prime (n)
  (if (<= n 1)
      nil
      (loop for i from 2 to (isqrt n)
            never (zerop (mod n i)))))
```

Timeline

1640

Pascaline



1840

Analytical Engine



1930

Zuse: Z3



1940

ENIAC



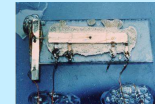
1950

TX-0



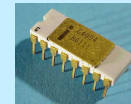
1960

IC



1970

Mikroprozessor



Hw

Sw

EP

Ada Lovelace



Arithm. Befehle

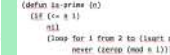


Assembler

Fortran Algol



Lisp



Entwicklungsprozess

Strukturierte
Program-
mierung

1960er: Die Gewinner sind

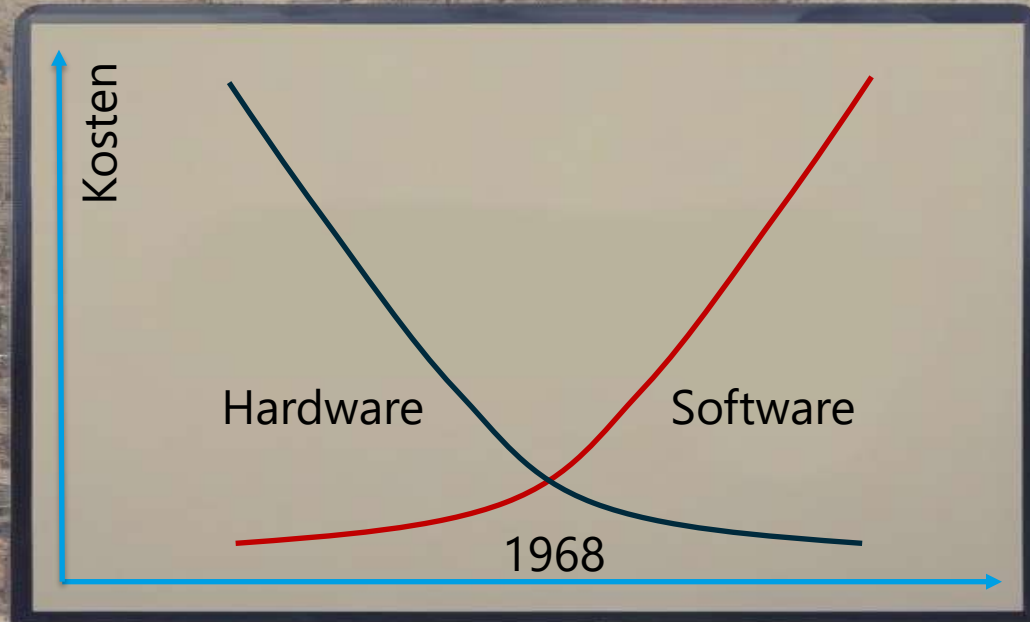
Fortran

Cobol

Assembler



1970er: 1. Software-Krise



Timeline

1640

Pascaline



1840

Analytical Engine



1930

Zuse: Z3



1940

ENIAC



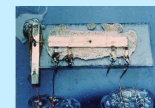
1950

TX-0



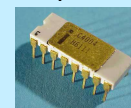
1960

IC



1970

Mikroprozessor



Hw

Sw

EP

KI

Ada Lovelace



Arithm. Befehle

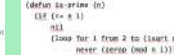


Assembler

Fortran Algol



Lisp



Strukturierte
Program-
mierung

Künstliche Intelligenz

Thema KI: Warum ist das bloß so schwierig für Computer?

› Computer können **gut** „**exakt**“

- ◆ rechnen: billionenfach schneller als Menschen
- ◆ Extrem schnell miteinander kommunizieren: milliardenfach schneller
- ◆ immens viele Daten exakt speichern, und **exakt** vergleichen

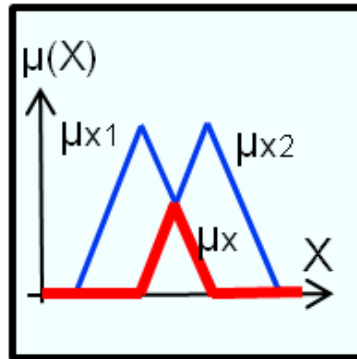
› Computer konnten **schlecht** “**ähnlich**“

- ◆ sehen und Dinge dabei erkennen
- ◆ hören und dabei Dinge erkennen
- ◆ **ähnliche** Dinge ausmachen und klassifizieren

› Menschliche Intelligenz zeichnet sich dadurch aus, eine **Hierarchie** von Konzepten zu verstehen

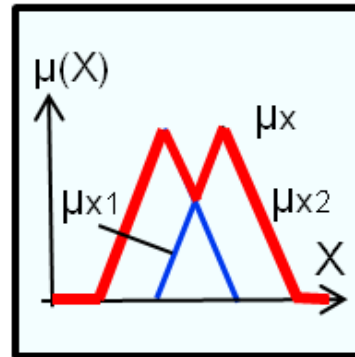
1960er: Fuzzy-Logic (Lotfi A. Zadeh)

UND-Verknüpfung min-Operator



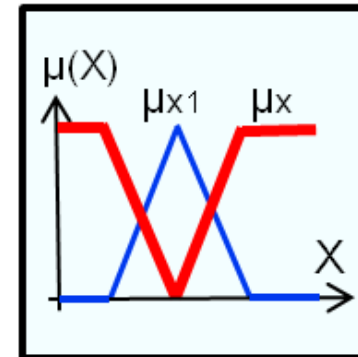
$$\begin{aligned}\mu_x(X) &= \min [\mu_{x1}(X), \mu_{x2}(X)] \\ &= (\mu_{x1} \cap \mu_{x2})(X)\end{aligned}$$

ODER-Verknüpfung max-Operator



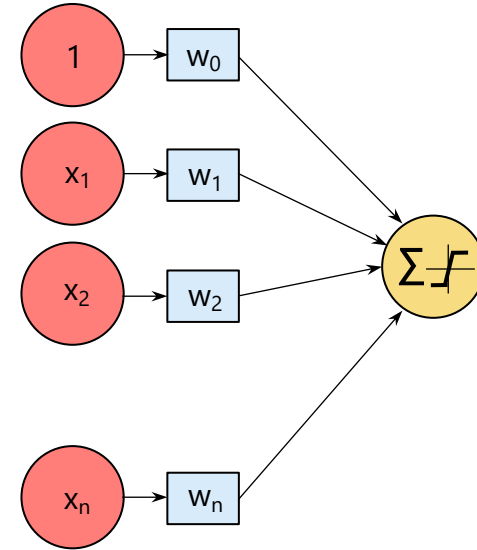
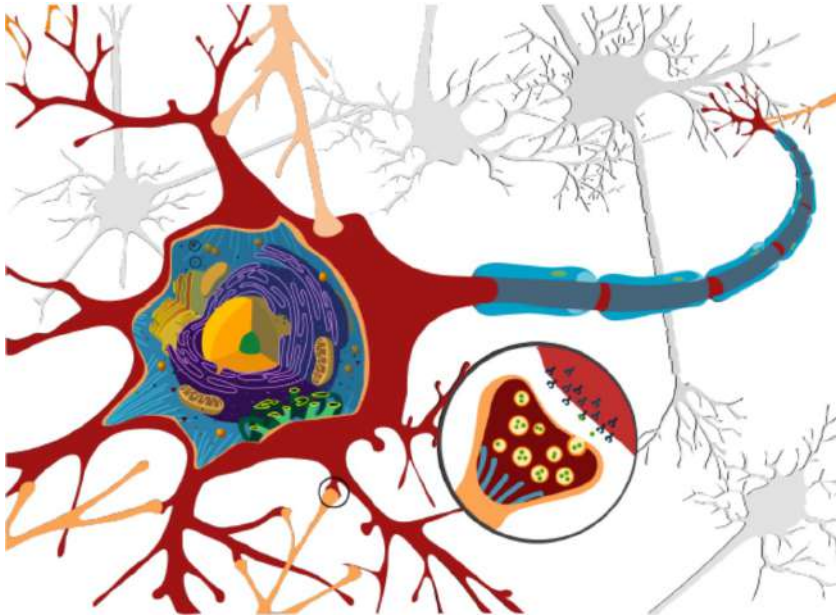
$$\begin{aligned}\mu_x(X) &= \max [\mu_{x1}(X), \mu_{x2}(X)] \\ &= (\mu_{x1} \cup \mu_{x2})(X)\end{aligned}$$

NICHT-Funktion Komplement-Operator



$$\mu_x(X) = 1 - \mu_{x1}(X)$$

1960er: Das Perzeptron (Frank Rosenblatt)



Hebb'sche Lernregel



1970er: Der 1. KI-Winter

Verarbeitung komplexerer neuronaler Netze?

Aus Hardware-Sicht unmöglich!

Timeline

1640

Pascaline



1840

Analytical Engine



1930

Zuse: Z3



1940

ENIAC



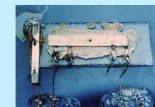
1950

TX-0



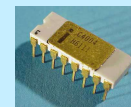
1960

IC



1970

Mikroprozessor



Hw

Sw

EP

KI

Ada Lovelace



Arithm. Befehle

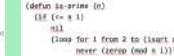


Assembler

Fortran Algol

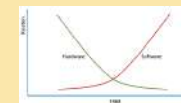


Lisp



Strukturierte
Program-
mierung

1. Softwarekrise



Perzeptron



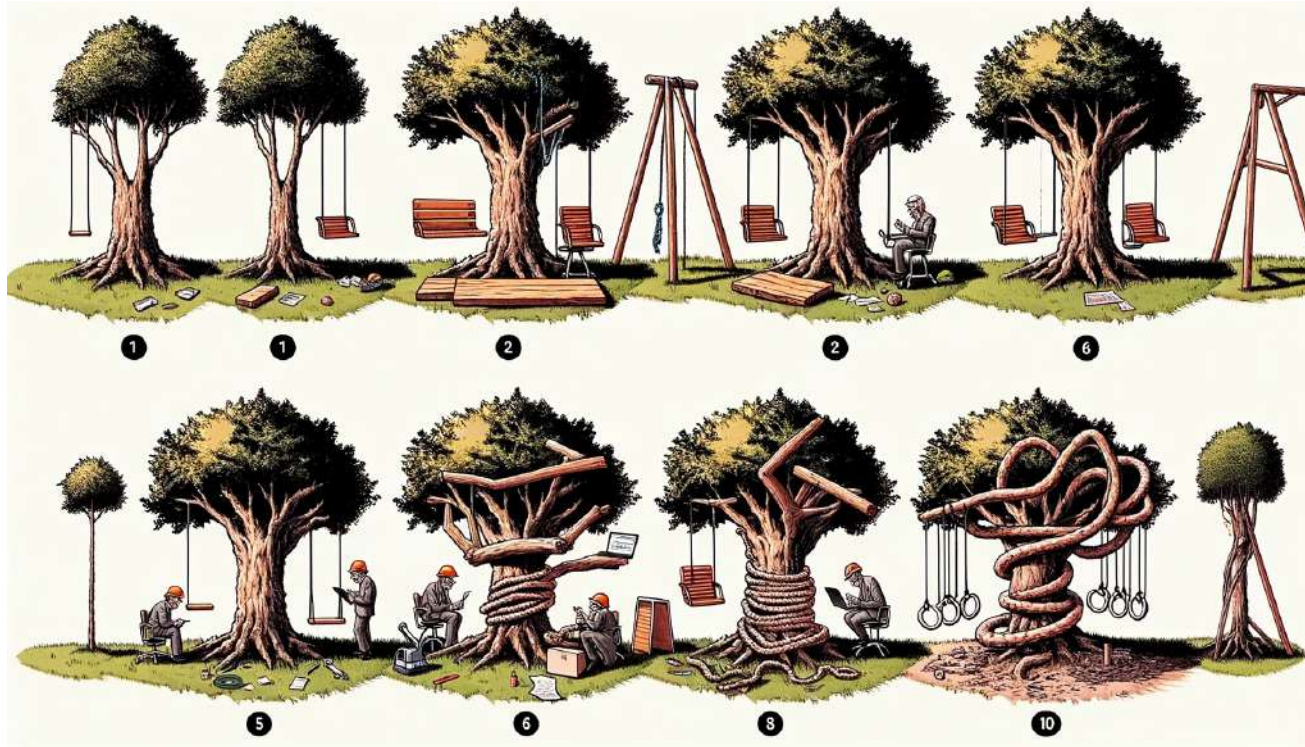
Fuzzylogik



1. KI-Winter



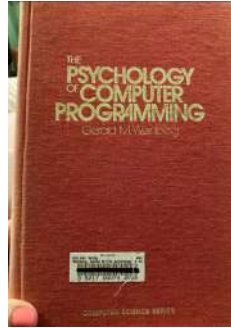
1970er: Warum ist Software nur so teuer?



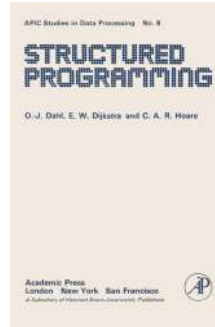
1970er: Software-Entwicklung wie ein Ingenieur



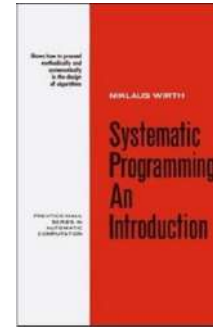
1970



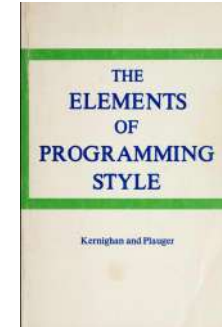
1971



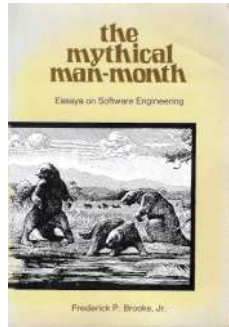
1972



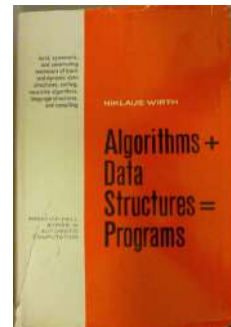
1973



1974



1975



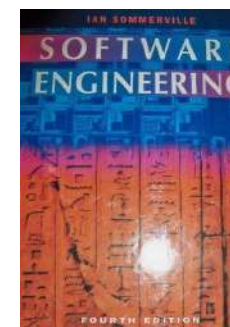
1976



1975



1978



1982

Die neue Kollegin ist da

1970er: Alternative KI-Forschung

- › **Modellierung der Welt**
- › **Expertensysteme:** Komplexe Entscheidungen
 - ◆ Regelbasierte Systeme: Prolog(!)
 - ◆ Fuzzylogik
 - ◆ Bayesianische Netze
 - ◆ Wissensrepräsentation: Frames, Schemata
 - ◆ Ontologien
- › **Genetische Algorithmen:** Komplexe Optimierungen
- › **Interaktion zwischen Menschen und Computern**

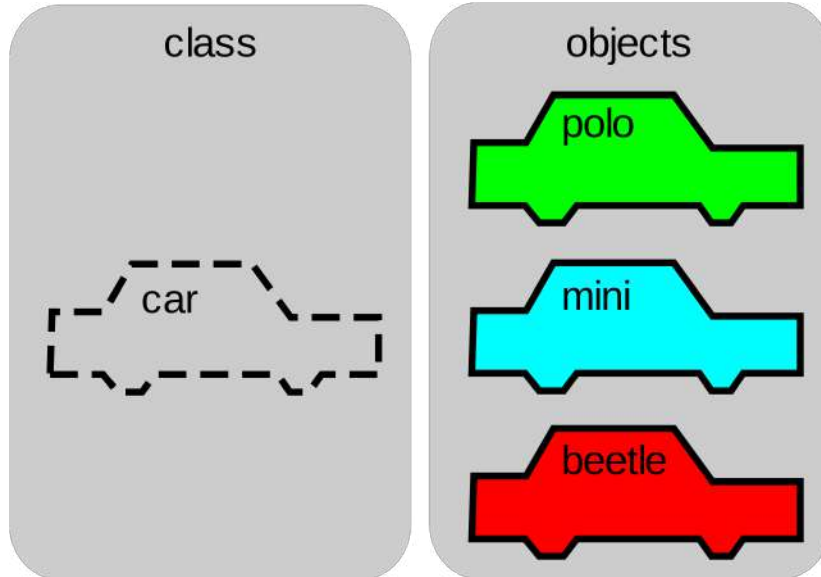
1970er: NLS von Doug Engelbart

.V2=1;10 .V=2;Editing Development
Innovations within the community have included development of a system
to track and report automatically the progress of documents through
production. Developments under consideration include an online "blue
pencil" system wherein an editor would mark video-displayed text with
revision lines and an author could later reject
.V2=1;11 .V=2;Printing Tools
With NLS you can print in a simple draft form, like galleys, or a format

oNLine System (NLS) SRI, US, 1960-70s
NLS was Doug Engelbart's system for word processing, hypertext, group
collaboration, and more. Terminals featured a monitor that could display
graphics in addition to text, a mouse, and a five-button chord keyset.

1970er: Objektorientierung (auch gut für GUIs)

Modellierung der Welt



1. OOP
2. OOD
3. OOA
4. OOM
5. OOSE
6. Booch
7. OMT

1. (Modula, C)
2. Simula
3. Smalltalk
4. C++
5. Objective C

1970er: Smalltalk (Kay, Ingalls, Goldberg)

```
isPrime: n
  ^(n <= 1)
    ifTrue: [false]
    ifFalse: [((2 to: (n sqrt) ceiling)
      allSatisfy: [:it | n \\ it ~= 0])].
```

1970er: Relationale Datenbanken (E. F. Codd), SQL



Schueler

SNR	Name	Vorname
6431	Euler	Leo
8151	Gauß	Carl
8152	Gödel	Kurt
7687	Klein	Felix

Kursbelegung

SNR	CNR
8151	13
6431	13
8151	24
8151	31

Kurs

CNR	Name	LK
13	I LK2	1
24	M LK	1
31	E LK	1
26	M GK	0



```
SELECT Schueler.Name, Kurs.Name FROM Schueler
JOIN Kursbelegung ON Schueler.SNR = Kursbelegung.SNR
JOIN Kurs ON Kursbelegung.CNR = Kurs.CNR
```

Die neue Kollegin ist da

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n -ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, comparison, join, retrieval language, predicate calculus, security, data integrity.

CC CATEGORIES: 3.70, 3.73, 3.75, 4.30, 4.32, 4.39

1. Relational Model and Normal Form

1.1. INTRODUCTION

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levin and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of data independence—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of data inconsistency which are expected to become troublesome even in non deductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for testing derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

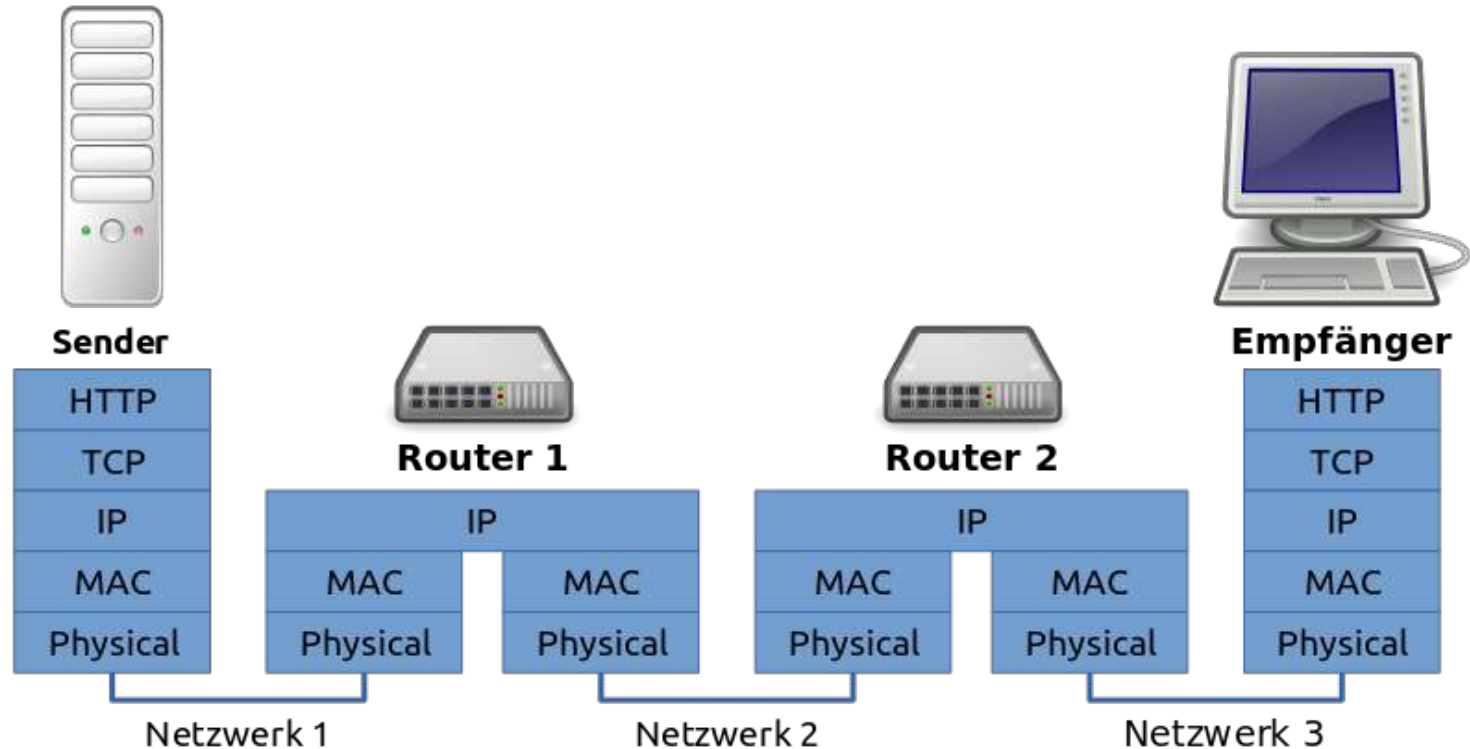
Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

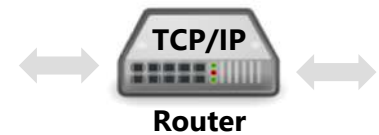
The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed without logically impairing some application programs is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of selections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems those dependencies are not clearly separable from one another.

1.2.1. Ordering Dependence. Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

1970er: Netzwerke, TCP/IP (Cerf, Kahn)



1970er: Unix (Thompson, Ritchie)



Timeline

1970

1980

1990

2000

Mikroprozessor



Netzwerke



Der PC



Die Maus



Pascal

C

SQL

Modula2

Smalltalk

```
isPrime: n
  ^n <= 1
  ifTrue: [false]
  ifFalse: [(0 to: n sqrt ceiling)
    do:[:i| n % i == 0] => false].
```

Software-Technik



Unix

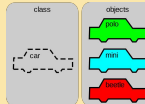


RDBs



TCP/IP

OOP



GUIs

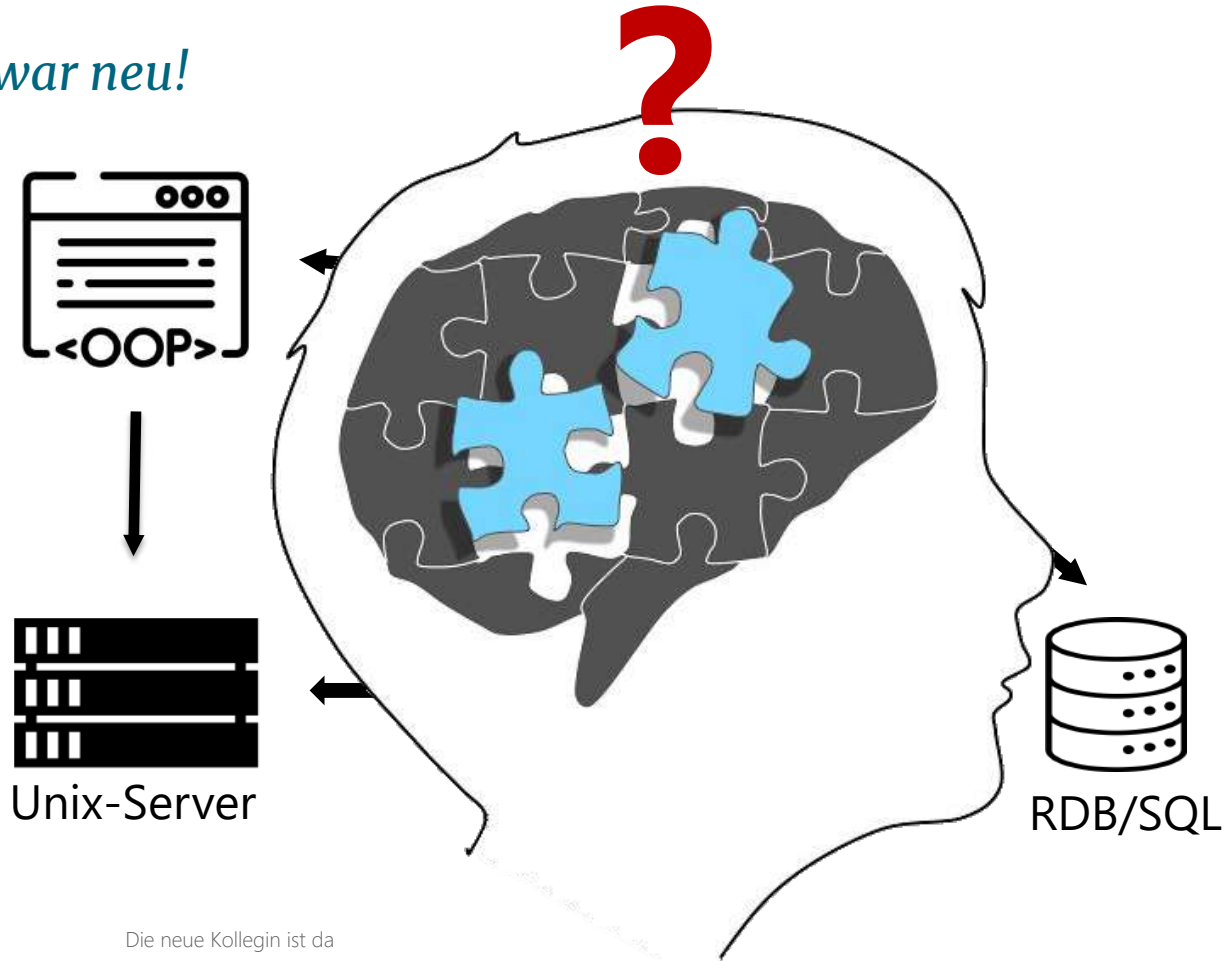


Genetische
Algorithmen

Wissens-
repräsen-
tation

Experten-
systeme

Zu viel war neu!



1980er: 2. Softwarekrise

A surreal digital landscape. In the foreground, a long, straight path made of wooden planks leads towards the horizon. On either side of the path are stacks of wooden crates or boxes. Several small figures of people are walking or sitting on the path and the crates. In the background, a large, glowing screen or monitor is the central focus. The screen displays the text "SECOND SOFTWARE CRISIS" at the top. Below this, there is a section titled "OPERATION: COUSIN" and another titled "in parentheses...". The screen also shows a large, stylized "C" logo and some text that appears to be a list of names or entities, including "resent action", "lionian 19", "1980", "1981", "1982", "1983", "1984", "1985", "1986", "1987", "1988", "1989", "1990", "1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023", "2024", "2025", "2026", "2027", "2028", "2029", "2030", "2031", "2032", "2033", "2034", "2035", "2036", "2037", "2038", "2039", "2040", "2041", "2042", "2043", "2044", "2045", "2046", "2047", "2048", "2049", "2050", "2051", "2052", "2053", "2054", "2055", "2056", "2057", "2058", "2059", "2060", "2061", "2062", "2063", "2064", "2065", "2066", "2067", "2068", "2069", "2070", "2071", "2072", "2073", "2074", "2075", "2076", "2077", "2078", "2079", "2080", "2081", "2082", "2083", "2084", "2085", "2086", "2087", "2088", "2089", "2090", "2091", "2092", "2093", "2094", "2095", "2096", "2097", "2098", "2099", "2100", "2101", "2102", "2103", "2104", "2105", "2106", "2107", "2108", "2109", "2110", "2111", "2112", "2113", "2114", "2115", "2116", "2117", "2118", "2119", "2120", "2121", "2122", "2123", "2124", "2125", "2126", "2127", "2128", "2129", "2130", "2131", "2132", "2133", "2134", "2135", "2136", "2137", "2138", "2139", "2140", "2141", "2142", "2143", "2144", "2145", "2146", "2147", "2148", "2149", "2150", "2151", "2152", "2153", "2154", "2155", "2156", "2157", "2158", "2159", "2160", "2161", "2162", "2163", "2164", "2165", "2166", "2167", "2168", "2169", "2170", "2171", "2172", "2173", "2174", "2175", "2176", "2177", "2178", "2179", "2180", "2181", "2182", "2183", "2184", "2185", "2186", "2187", "2188", "2189", "2190", "2191", "2192", "2193", "2194", "2195", "2196", "2197", "2198", "2199", "2200", "2201", "2202", "2203", "2204", "2205", "2206", "2207", "2208", "2209", "2210", "2211", "2212", "2213", "2214", "2215", "2216", "2217", "2218", "2219", "2220", "2221", "2222", "2223", "2224", "2225", "2226", "2227", "2228", "2229", "2230", "2231", "2232", "2233", "2234", "2235", "2236", "2237", "2238", "2239", "2240", "2241", "2242", "2243", "2244", "2245", "2246", "2247", "2248", "2249", "2250", "2251", "2252", "2253", "2254", "2255", "2256", "2257", "2258", "2259", "2260", "2261", "2262", "2263", "2264", "2265", "2266", "2267", "2268", "2269", "2270", "2271", "2272", "2273", "2274", "2275", "2276", "2277", "2278", "2279", "2280", "2281", "2282", "2283", "2284", "2285", "2286", "2287", "2288", "2289", "2290", "2291", "2292", "2293", "2294", "2295", "2296", "2297", "2298", "2299", "2300", "2301", "2302", "2303", "2304", "2305", "2306", "2307", "2308", "2309", "2310", "2311", "2312", "2313", "2314", "2315", "2316", "2317", "2318", "2319", "2320", "2321", "2322", "2323", "2324", "2325", "2326", "2327", "2328", "2329", "2330", "2331", "2332", "2333", "2334", "2335", "2336", "2337", "2338", "2339", "2340", "2341", "2342", "2343", "2344", "2345", "2346", "2347", "2348", "2349", "2350", "2351", "2352", "2353", "2354", "2355", "2356", "2357", "2358", "2359", "2360", "2361", "2362", "2363", "2364", "2365", "2366", "2367", "2368", "2369", "2370", "2371", "2372", "2373", "2374", "2375", "2376", "2377", "2378", "2379", "2380", "2381", "2382", "2383", "2384", "2385", "2386", "2387", "2388", "2389", "2390", "2391", "2392", "2393", "2394", "2395", "2396", "2397", "2398", "2399", "2400", "2401", "2402", "2403", "2404", "2405", "2406", "2407", "2408", "2409", "2410", "2411", "2412", "2413", "2414", "2415", "2416", "2417", "2418", "2419", "2420", "2421", "2422", "2423", "2424", "2425", "2426", "2427", "2428", "2429", "2430", "2431", "2432", "2433", "2434", "2435", "2436", "2437", "2438", "2439", "2440", "2441", "2442", "2443", "2444", "2445", "2446", "2447", "2448", "2449", "2450", "2451", "2452", "2453", "2454", "2455", "2456", "2457", "2458", "2459", "2460", "2461", "2462", "2463", "2464", "2465", "2466", "2467", "2468", "2469", "2470", "2471", "2472", "2473", "2474", "2475", "2476", "2477", "2478", "2479", "2480", "2481", "2482", "2483", "2484", "2485", "2486", "2487", "2488", "2489", "2490", "2491", "2492", "2493", "2494", "2495", "2496", "2497", "2498", "2499", "2500", "2501", "2502", "2503", "2504", "2505", "2506", "2507", "2508", "2509", "2510", "2511", "2512", "2513", "2514", "2515", "2516", "2517", "2518", "2519", "2520", "2521", "2522", "2523", "2524", "2525", "2526", "2527", "2528", "2529", "2530", "2531", "2532", "2533", "2534", "2535", "2536", "2537", "2538", "2539", "2540", "2541", "2542", "2543", "2544", "2545", "2546", "2547", "2548", "2549", "2550", "2551", "2552", "2553", "2554", "2555", "2556", "2557", "2558", "2559", "2560", "2561", "2562", "2563", "2564", "2565", "2566", "2567", "2568", "2569", "2570", "2571", "2572", "2573", "2574", "2575", "2576", "2577", "2578", "2579", "2580", "2581", "2582", "2583", "2584", "2585", "2586", "2587", "2588", "2589", "2590", "2591", "2592", "2593", "2594", "2595", "2596", "2597", "2598", "2599", "2600", "2601", "2602", "2603", "2604", "2605", "2606", "2607", "2608", "2609", "2610", "2611", "2612", "2613", "2614", "2615", "2616", "2617", "2618", "2619", "2620", "2621", "2622", "2623", "2624", "2625", "2626", "2627", "2628", "2629", "26

A photograph of a snowy forest path. The path is covered in deep snow with some tracks. The trees on either side are heavily laden with snow, their branches drooping. In the distance, a bright light source, possibly the sun, creates a strong glow and lens flare effect, illuminating the scene. The overall atmosphere is cold and serene.

1980er: 2. KI-Winter

Lösung für komplexe Probleme?
Erwartungen wurden erneut enttäuscht!

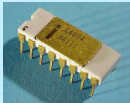
Timeline

1970

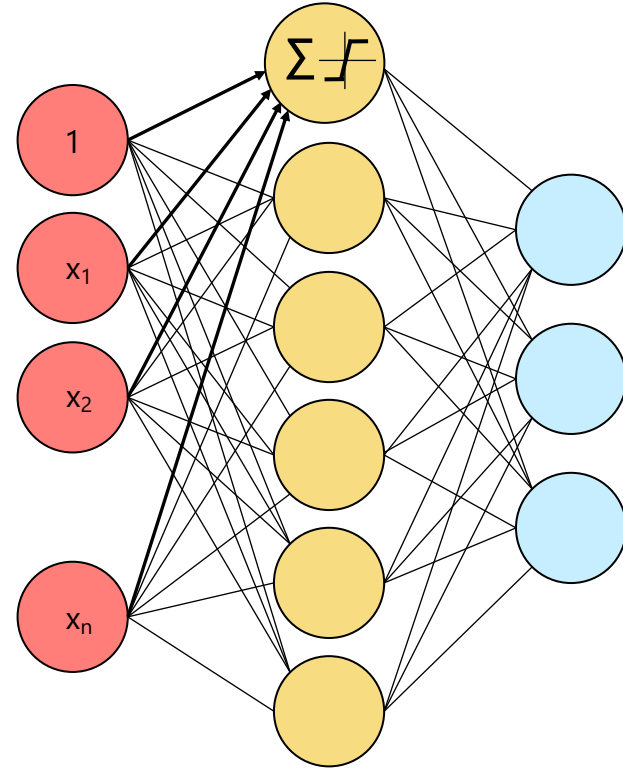
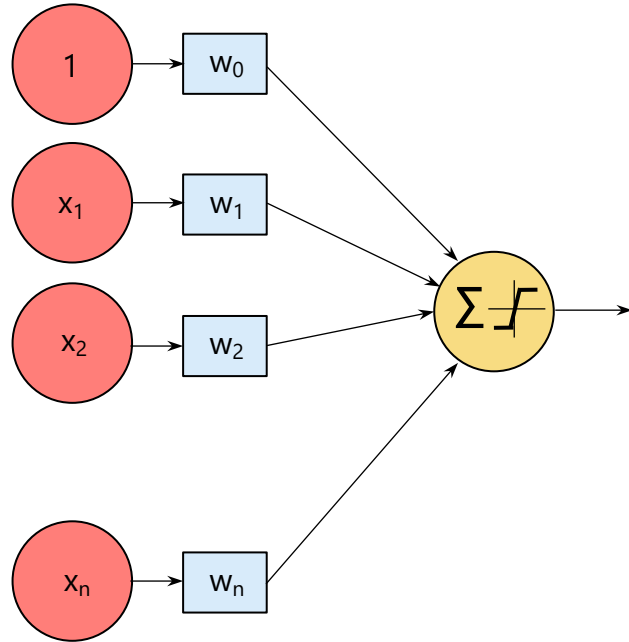
1980

1990

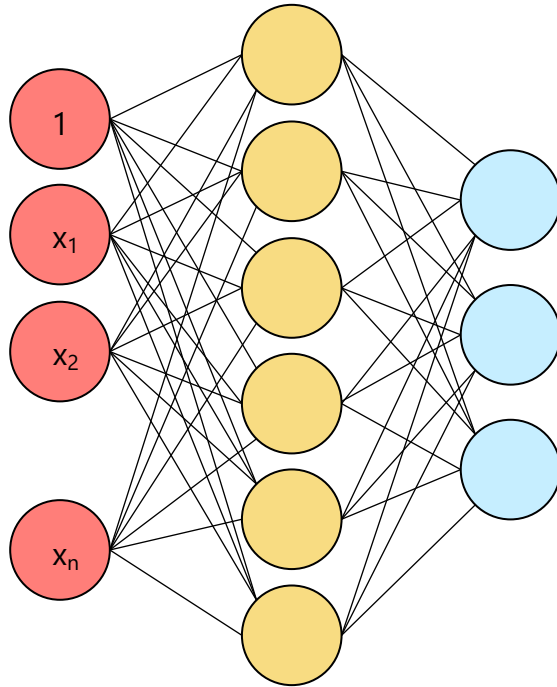
2000

Mikroprozessor	Netzwerke	Der PC	Die Maus
			
Pascal	C SQL Modula2	Smalltalk	C++
		<pre>isPrime: n ^n <= 1 ifTrue: [false] ifFalse: [(0 to: n sqrt ceiling) at: 2 step: 1 do: i n % i == 0]]</pre>	
Software-Technik	Unix	RDBs	TCP/IP
			
		OOP	GUIs
			
			2. Softwarekrise
			
Genetische Algorithmen	Wissens-repräsentation	Experten-systeme	2. KI-Winter
			

1980er: Backpropagation (Werbos, Rumelhart, Hinton, Williams)

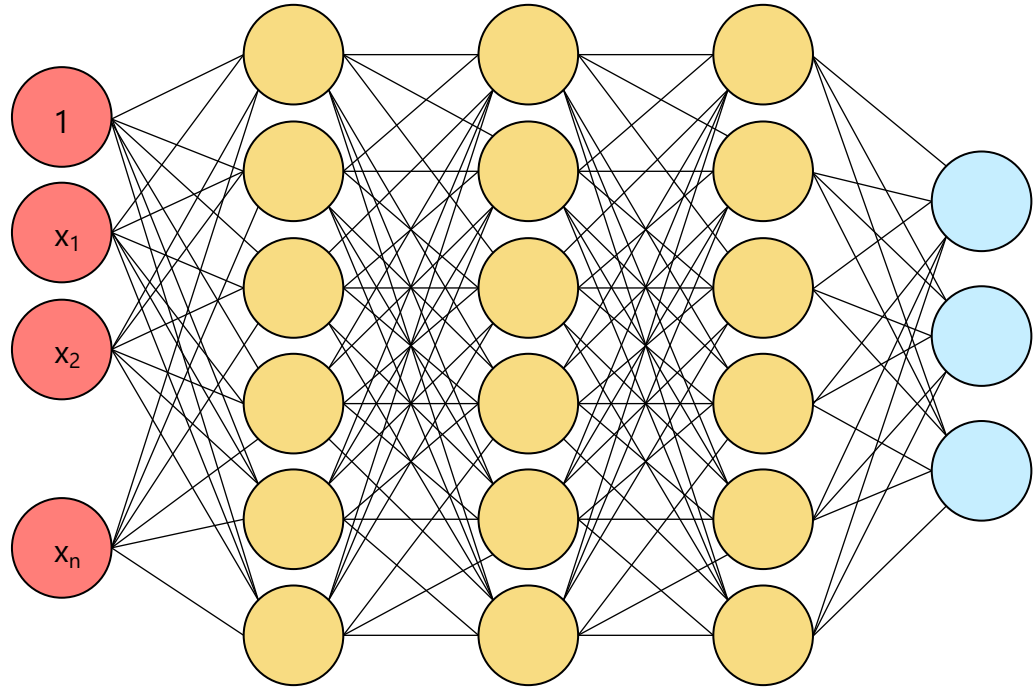


1990er: Deep Learning



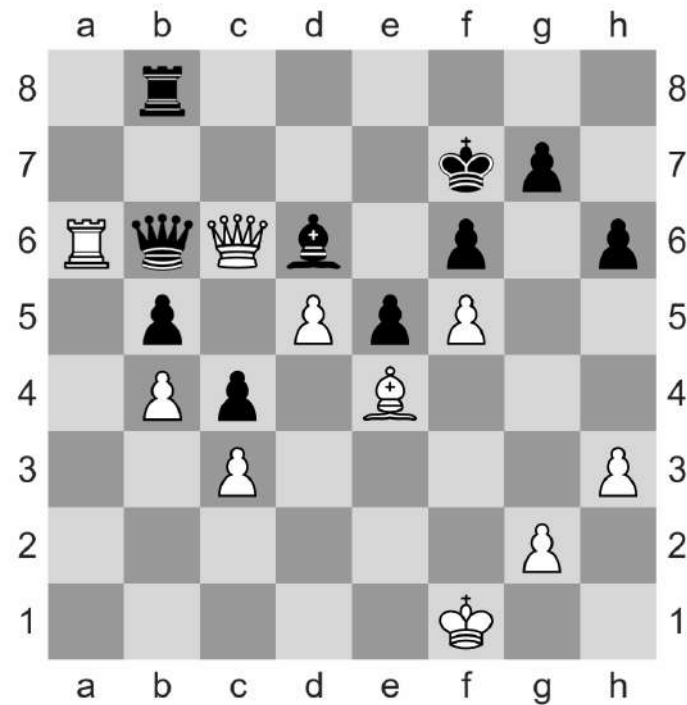
einfaches NN

Die neue Kollegin ist da



tiefes NN

1997: Deep Blue



Timeline

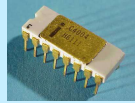
1970

1980

1990

2000

Mikroprozessor



Netzwerke



Der PC



Die Maus



Pascal

C

SQL

Modula2

Smalltalk

C++

```
isPrime: a
  ^in 2..a
  ifTrue: [false]
  ifFalse: [(2 to: a sqrt ceiling)
    do: [p | a % p == 0] => false].
```

Software-Technik



Unix

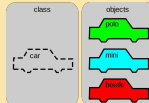


RDBs



TCP/IP

OOP



GUIs



2. Softwarekrise



Genetische Algorithmen

Wissens-repräsen-tation

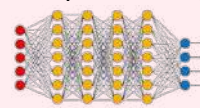
Experten-systeme

2. KI-Winter



Back propagation

"Deep" Learning



Deep Blue





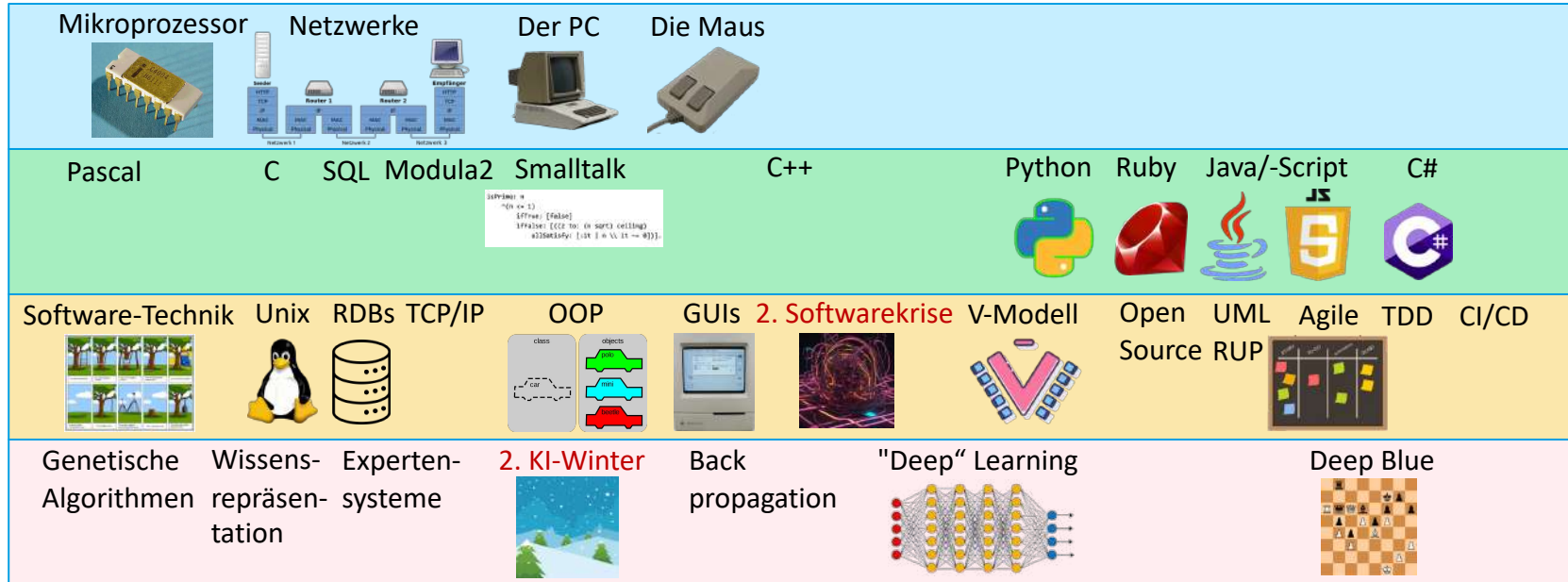
Timeline

1970

1980

1990

2000

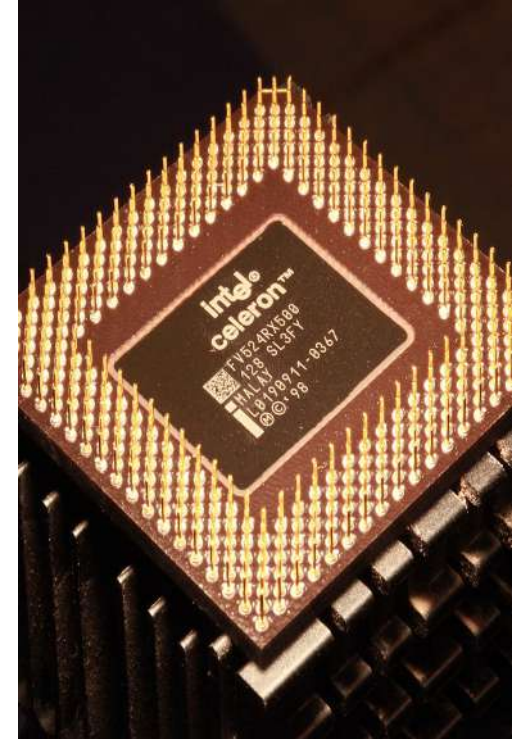


1990er: World Wide Web

Sir Tim Berners-Lee



1990er: Der moderne PC im Job und zu Hause



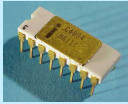
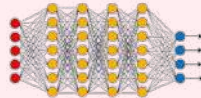
Timeline

1970

1980

1990

2000

Mikroprozessor	Netzwerke	Der PC	Die Maus	Internet-Boom	PC, Flachbildschirm							
												
Pascal	C	SQL	Modula2	Smalltalk	C++	Python	Ruby	Java/-Script	C#			
				<pre>jsprimo: a ~in: ex 1: {True: [false]} {False: [[cc to: in surc ceiling] at2500000: [-10 1 0 10 10 0]]} </pre>								
Software-Technik	Unix	RDBs	TCP/IP	OOP	GUIs	2. Softwarekrise	V-Modell	Open Source	UML	Agile	TDD	CI/CD
												
Genetische Algorithmen	Wissens-repräsen-tation	Experten-systeme	2. KI-Winter	Back propagation	"Deep" Learning						Deep Blue	
												

2010er: Das Smartphone



2010er: Leistungsfähige Multicore-CPUs und GPUs



2020er: Heutige Rechenzentren, Cloud Computing





2020er:
*Der erste kommerzielle
Quantencomputer*

Jahr	Technik	Beispiel	Leistung	Speicher
1642	Mechanik	Pascaline, Blaise Pascal		
1930	Relais	Konrad Zuse: Z3	1	176
1940	Vakuumröhren	ENIAC: Erster digitaler Computer	100	2000
1950	Transistoren	Point-Contact Transistor, Bell	1.000	10.000
1960	Integrierte Schaltkreise	µA702, Fairchild Semiconductor	10.000	100.000
1970	Mikroprozessoren	Intel 4004, 4Bit, 740 kHz	1.000.000	1.000.000
2000	Smartphones	iPhone, Apple	1.000.000.000.000	10.000.000.000
2010	Cloud-Computing	AWS, Azure, GCP	quasi unbegrenzt	quasi unbegrenzt
2020	Quanten-Computing	IBM Q System One	nicht vergleichbar	nicht vergleichbar

100Jahrzehnt

Timeline

2000

Multicore
CPUs
GPUs

Megazentren



2010

Smartphone



2020

Quanten Computing



2030

2000er: Scala



```
def isPrime(n: Int): Boolean = {  
    if (n <= 1) false  
    else !(2 to math.sqrt(n).toInt).exists(n % _ == 0)  
}
```

2010er: Kotlin

```
fun isPrime(n: Int) =  
    if (n <= 1) false  
    else (2..sqrt(n)).none { it divides n }
```



2020er: Comment Driven Development (CDD)



```
/**  
 * Testet eine Zahl auf Primzahleigenschaft in Python  
 */
```

```
def is_prime(number):  
    if number <= 1: return False  
    for i in range(2, int(sqrt(number))+1):  
        if number % i == 0:  
            return False  
    return True
```

Wie rede ich mit Maschinen?

Level 1

1950er: Maschinen-Code

```
11100000 00000010
11110000 00000010
11100000 00000001
11110000 00000010
10100010 00000010
11001010
11110000 00000010
11101100 00010000 00000000
11110000 00000010
11100000 00000001
11110000 00000010
10010000 11110000
00111000
01100000
00011000
01100000
```

Level 2

1960er: Assembler

```
start:
    LDA number
    JSR is_prime
    BCS not_prime
is_prime:
    CPX #2
    BEQ prime_done
    CPX #1
    BEQ prime_done
    LDX #2
check_divisible:
    DEX
    BEQ prime_done
    CPX number
    BEQ prime_done
    CPX #1
    BEQ prime_done
    BCC check_divisible
    SEC
    RTS
prime_done:
    CLC
    RTS
```

Level 3

1960er: Cobol

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 Num PIC 9(5).
01 IsPrime PIC X.

PROCEDURE DIVISION.
    Accept-Num.
        Perform Prime-Number-Test.
        STOP RUN.

    Accept-Num.
        ACCEPT Num.

    Prime-Number-Test.
        IF Num <= 1
            MOVE "N" TO IsPrime
        ELSE
            PERFORM Check-Divisors UNTIL Num
            <= Sqrt(Num)
            IF IsPrime IS INITIAL
                MOVE "Y" TO IsPrime
            END-IF.
        END-IF.

    Check-Divisors.
        DIVIDE Num BY Num - 1 GIVING
    Remainder
        IF Remainder = 0
            MOVE "N" TO IsPrime
        END-IF
        SUBTRACT 1 FROM Num.
```

Level 3

1960er: Fortran I

```
      READ(5,2) NUM
2     FORMAT(I)
      I = 2
3     IF(I**2-NUM) 4,5,5
4     PRINT 6
      STOP
5     IF(I-1) 6,7,7
6     I = I + 1
      GOTO 3
7     PRINT 8
      STOP
8     FORMAT(1H1, ' ', I5)
      END
```

Level 3

1980er: Python

```
def is_prime(number):
    if number <= 1: return False
    for i in range(2, int(sqrt(number))+1):
        if number % i == 0:
            return False
    return True
```

Level 4

2000er: Scala

```
def isPrime(n: Int): Boolean = {
    if (n <= 1) false
    else !(2 to math.sqrt(n).toInt).exists(n % _ == 0)
}
```

Level 4

2010er: Kotlin

```
fun isPrime(n: Int) =
    if (n <= 1) false
    else (2..sqrt(n)).none { it divides n }
```

Level 5

2020er: Comment Driven Development (CDD)

```
/**
 * Testet eine Zahl auf Primzahleigenschaft
 */
```

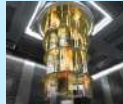
Timeline

2000

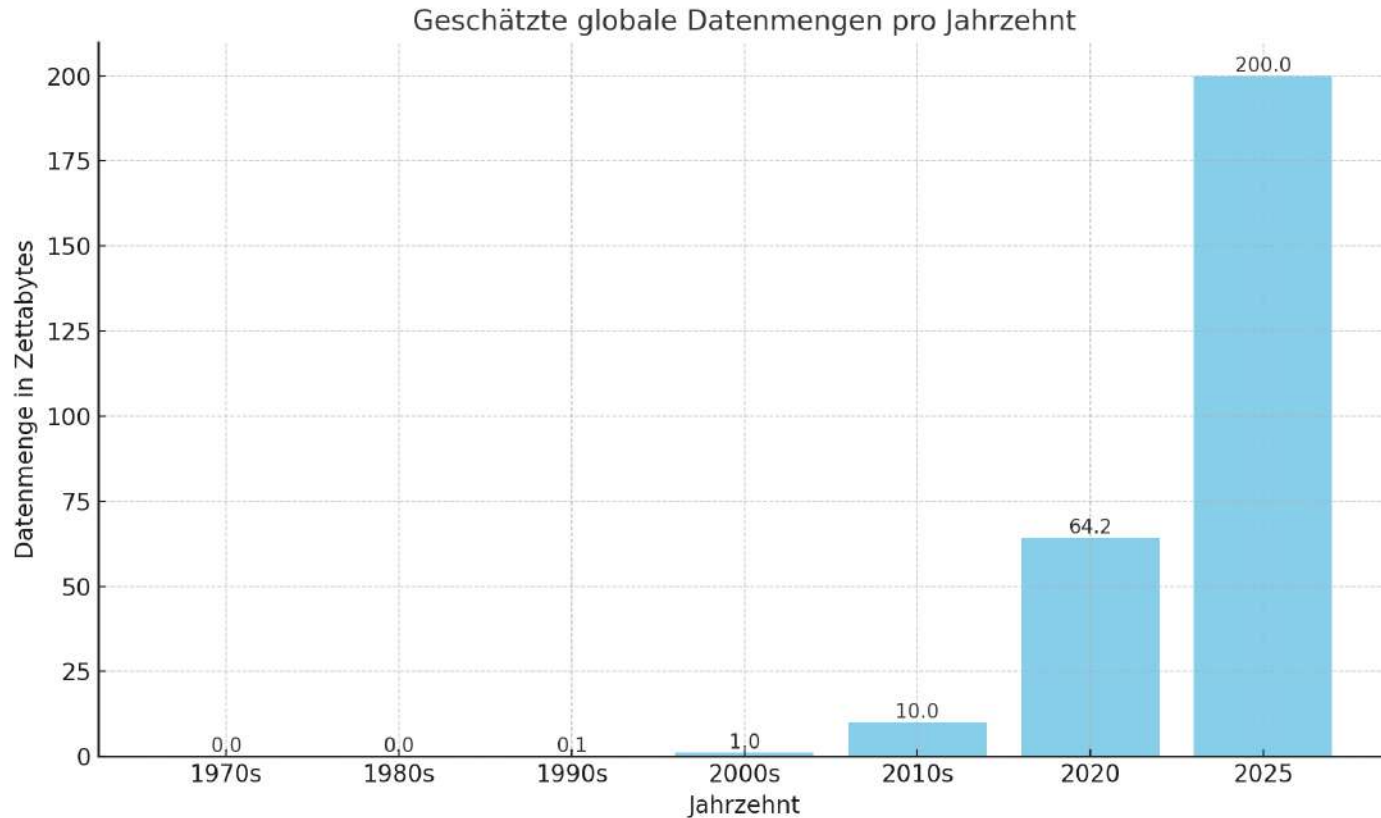
2010

2020

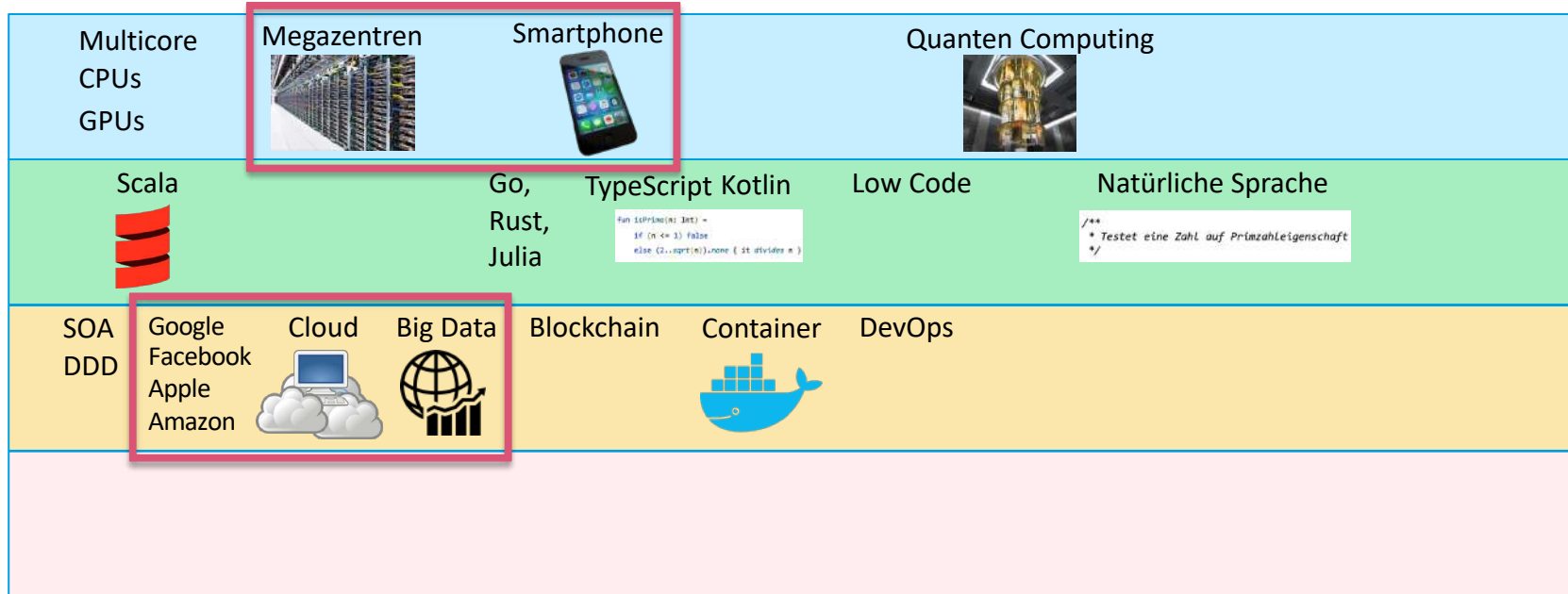
2030

Multicore CPUs GPUs	Megazentren 	Smartphone 	Quanten Computing 		
Scala 	Go, Rust, Julia		TypeScript Kotlin <pre>fun isPrime(n: Int) = if (n <= 1) false else (2..n/2).none { it divides n }</pre>	Low Code	Natürliche Sprache <pre>/** * Testet eine Zahl auf Primzahleigenschaft */</pre>

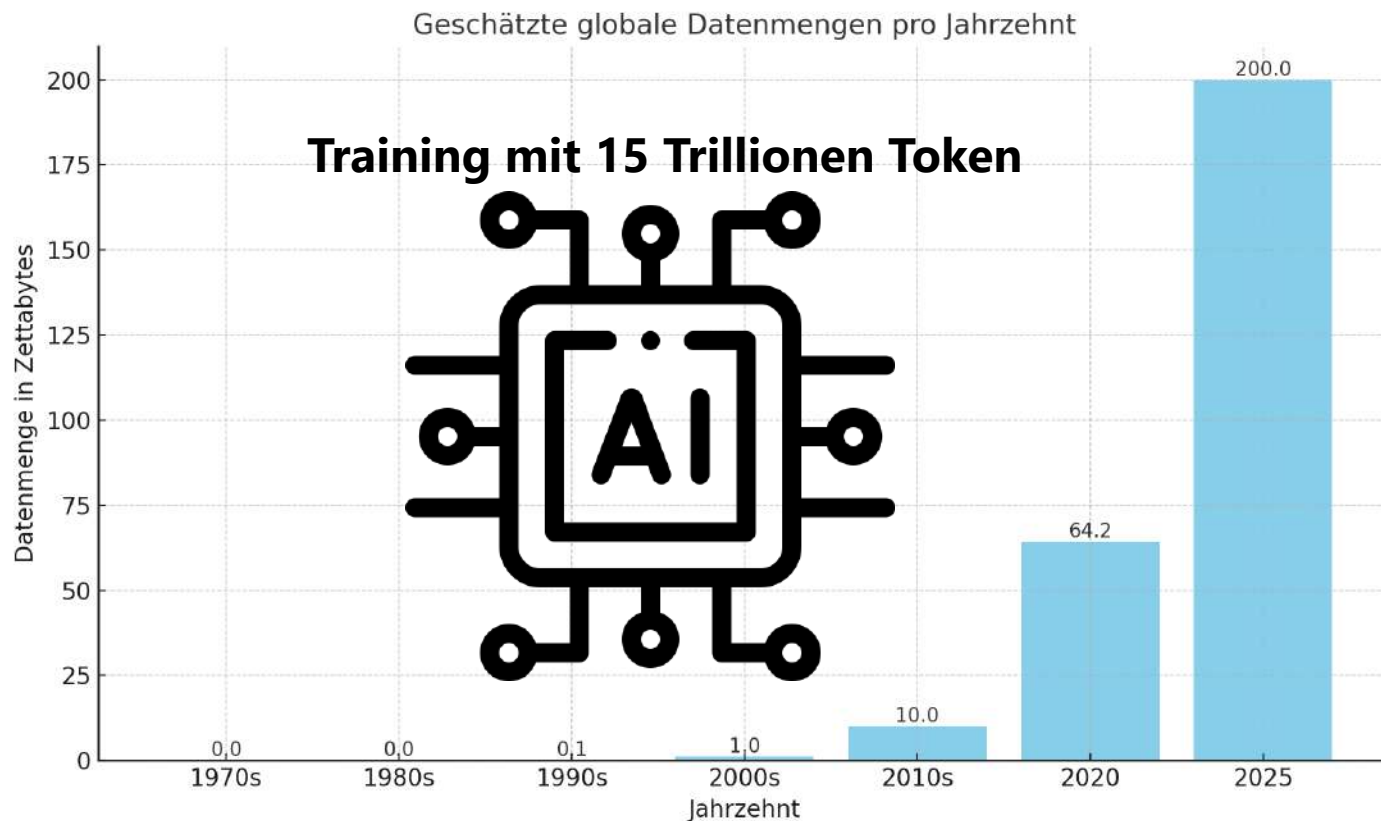
2010er: BigData



Timeline



2010er: BigData



2006: Unsupervised pre-training

A fast learning algorithm for deep belief nets *

Geoffrey E. Hinton and Simon Osindero

Department of Computer Science University of Toronto
10 Kings College Road
Toronto, Canada M5S 3G4
{hinton, osindero}@cs.toronto.edu

Yee-Whye Teh

Department of Computer Science
National University of Singapore
3 Science Drive 3, Singapore, 117543
tehyw@comp.nus.edu.sg

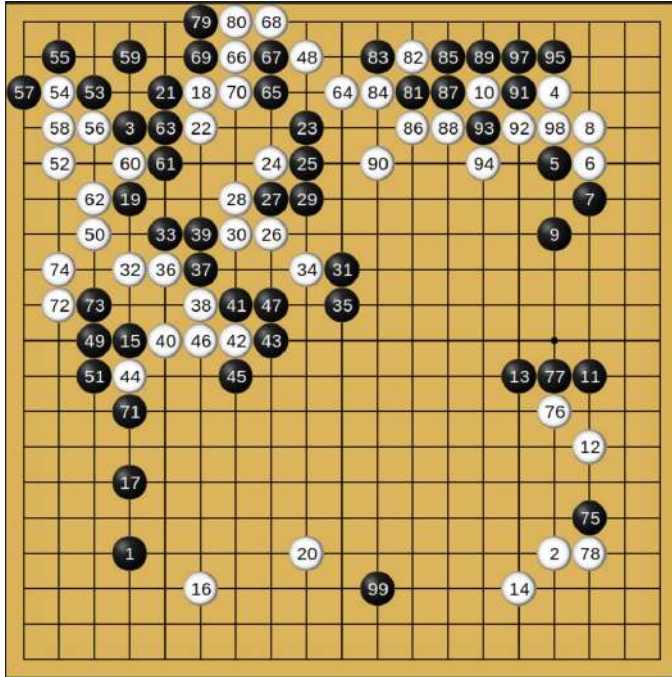
Abstract

We show how to use “complementary priors” to eliminate the explaining away effects that make inference difficult in densely-connected belief nets that have many hidden layers. Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory. The fast, greedy algorithm is used to initialize a slower learning procedure that fine-tunes the weights using a contrastive version of the wake-sleep algorithm. After fine-tuning, a network with three hidden layers forms a very good generative model of the joint distribution of handwritten digit images and their labels. This generative model gives

remaining hidden layers form a directed acyclic graph that converts the representations in the associative memory into observable variables such as the pixels of an image. This hybrid model has some attractive features:

1. There is a fast, greedy learning algorithm that can find a fairly good set of parameters quickly, even in deep networks with millions of parameters and many hidden layers.
2. The learning algorithm is unsupervised but can be applied to labeled data by learning a model that generates both the label and the data.
3. There is a fine-tuning algorithm that learns an excellent generative model which outperforms discriminative methods on the MNIST database of hand-written digits.

2016 : AlphaGo siegt gegen Lee Sedol



- überwachtes Lernen
- unüberwachtes Lernen
- Deep Neural Networks
 - Policy Network
 - Value Network
- Reinforcement Learning

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez*[†]
University of Toronto
aidan@cs.toronto.edu

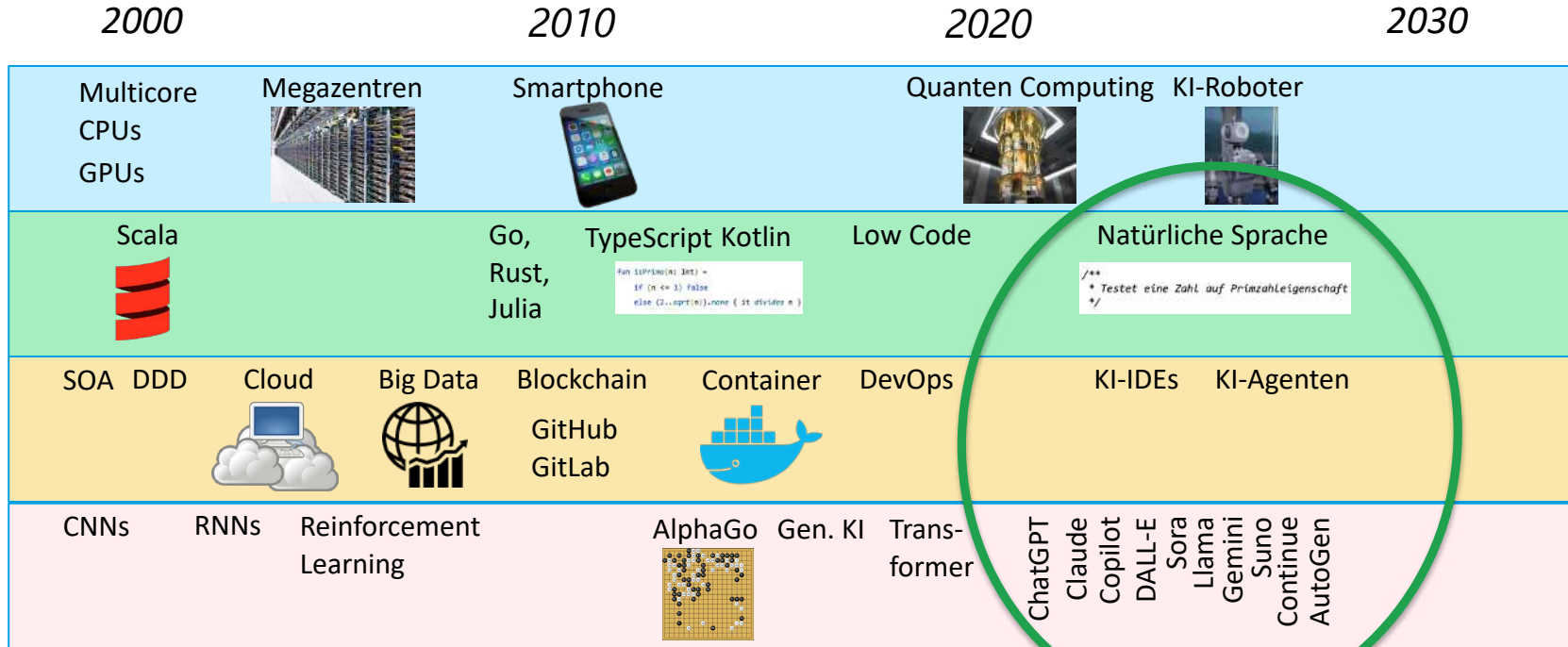
Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin*[‡]
illia.polosukhin@gmail.com

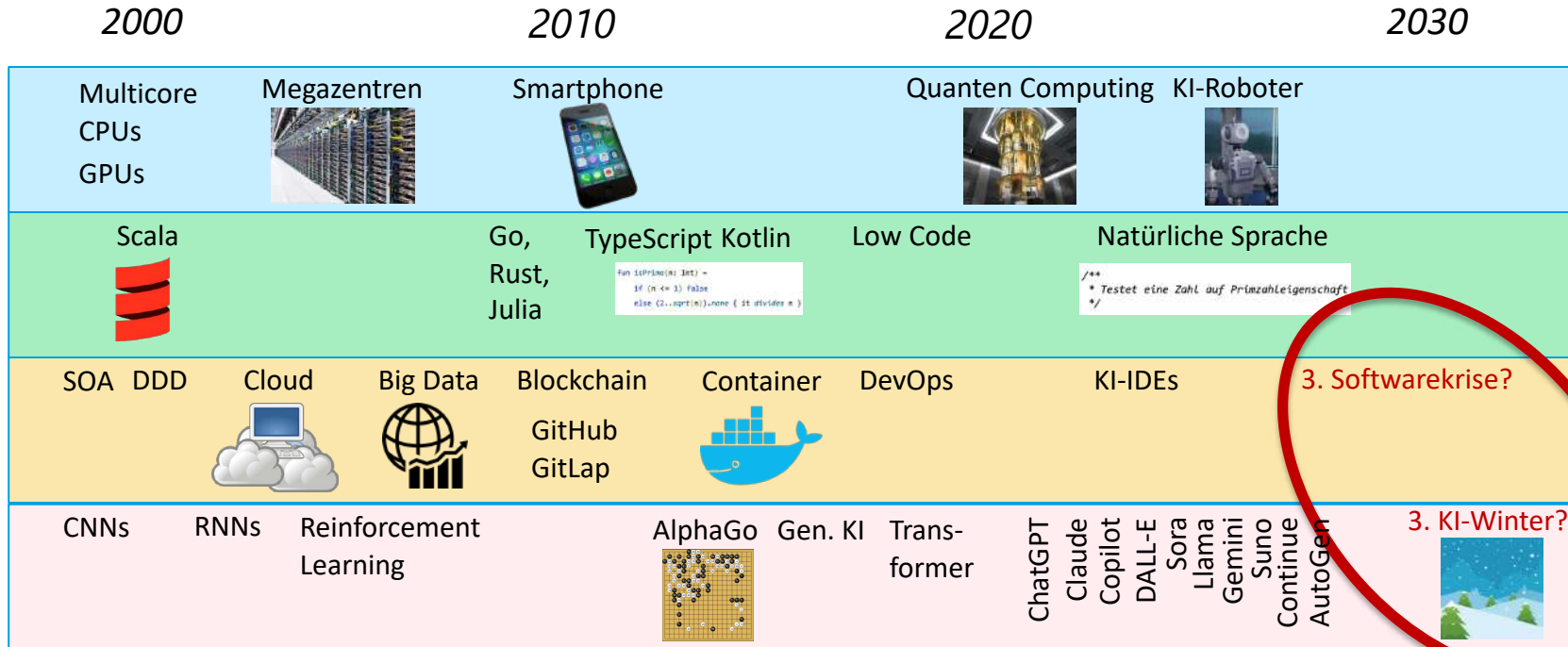
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions.

Timeline



Timeline



KAN: Kolmogorov–Arnold Networks

Ziming Liu^{1,4*} Yixuan Wang² Sachin Vaidya¹ Fabian Ruehle^{3,4}
James Halverson^{3,4} Marin Soljačić^{1,4} Thomas Y. Hou² Max Tegmark^{1,4}

¹ Massachusetts Institute of Technology

² California Institute of Technology

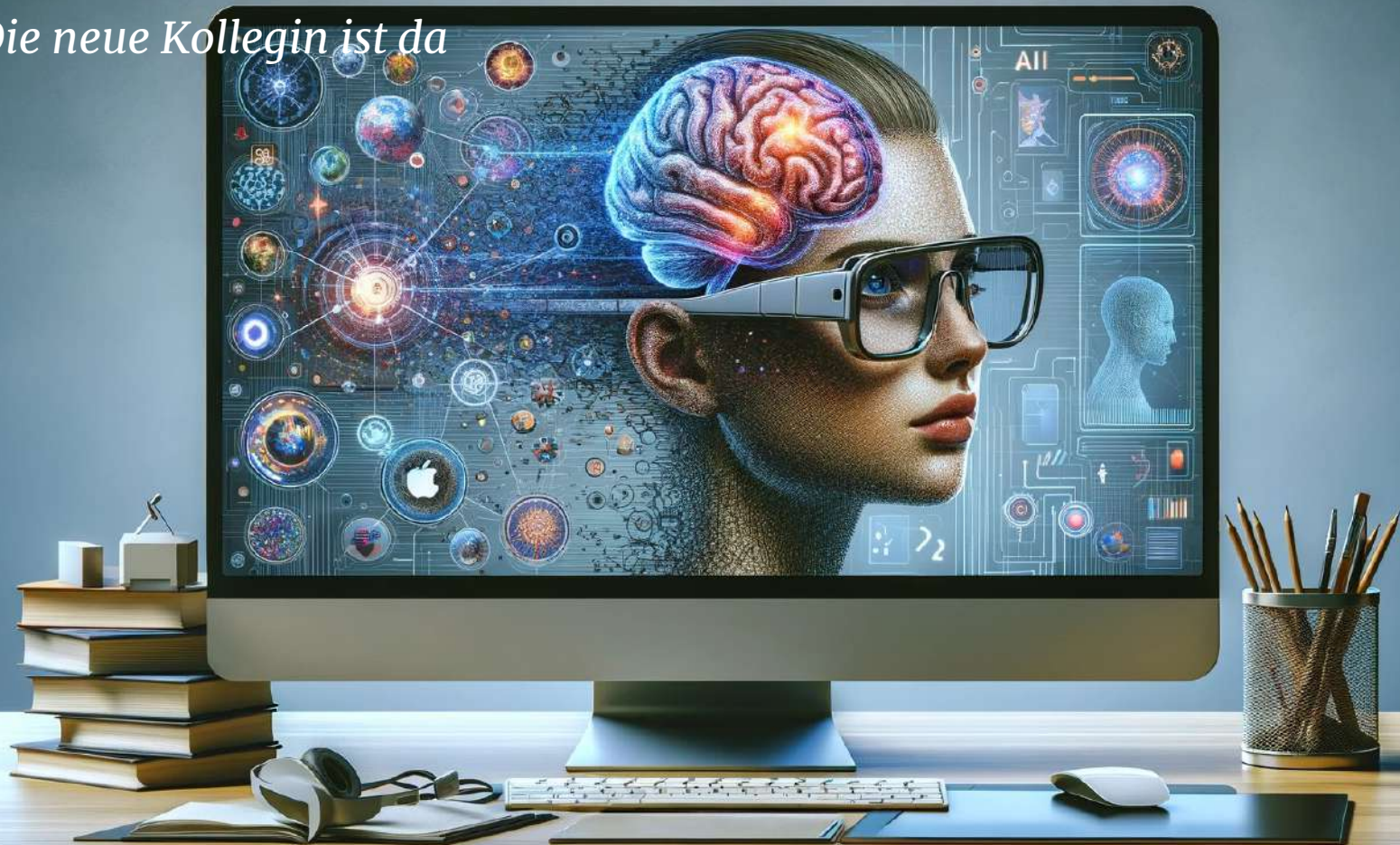
³ Northeastern University

⁴ The NSF Institute for Artificial Intelligence and Fundamental Interactions

Abstract

Inspired by the Kolmogorov–Arnold representation theorem, we propose Kolmogorov–Arnold Networks (KANs) as promising alternatives to Multi-Layer Perceptrons (MLPs). While MLPs have *fixed* activation functions on *nodes* (“neurons”), KANs have *learnable* activation functions on *edges* (“weights”). KANs have no linear weights at all – every weight parameter is replaced by a univariate function parametrized as a spline. We show that this seemingly simple change makes KANs outperform MLPs in terms of accuracy and interpretability. For accuracy, much smaller KANs can achieve comparable or better accuracy than much larger MLPs in data fitting and PDE solving. Theoretically and empirically, KANs possess faster neural scaling laws than MLPs. For interpretability, KANs

Die neue Kollegin ist da



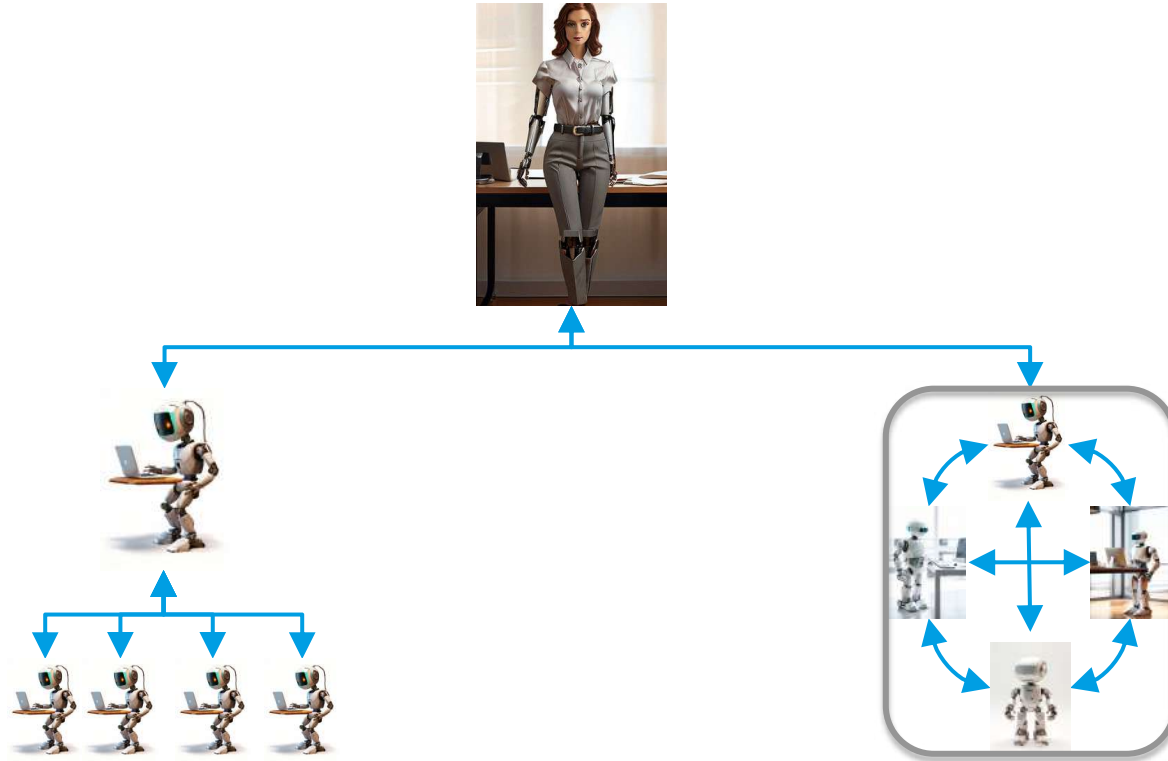
KI – Werkzeug oder Kollegin?



Die neue Kollegin ist da



Die neuen Kollegin steht für ein Team neuer Kolleginnen



Die neue Kollegin ist da

Die neue Kollegin ist da



Die neue Kollegin ist da

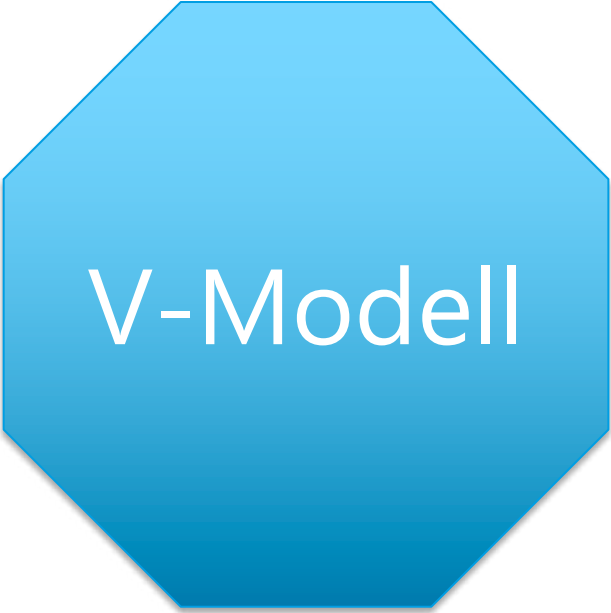
Die neuen Kollegen – Hype oder bleibende Innovation?



Die neue Kollegin ist da

Wie arbeiten Menschen zusammen?





V-Modell



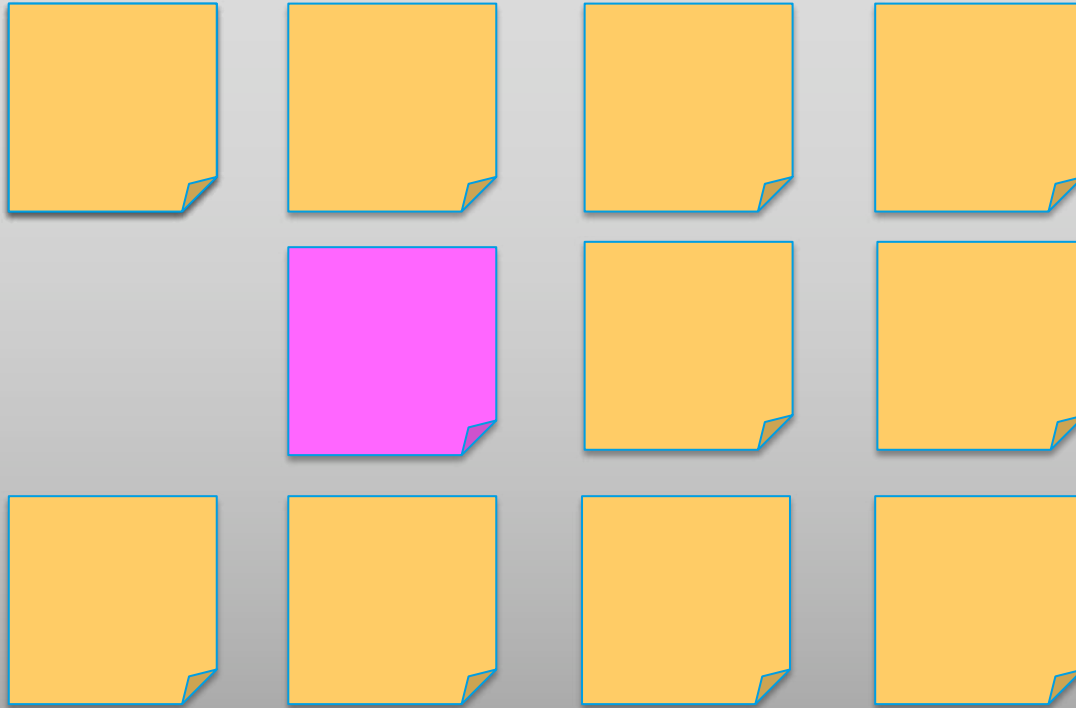
UML

Klassisch: Analyse



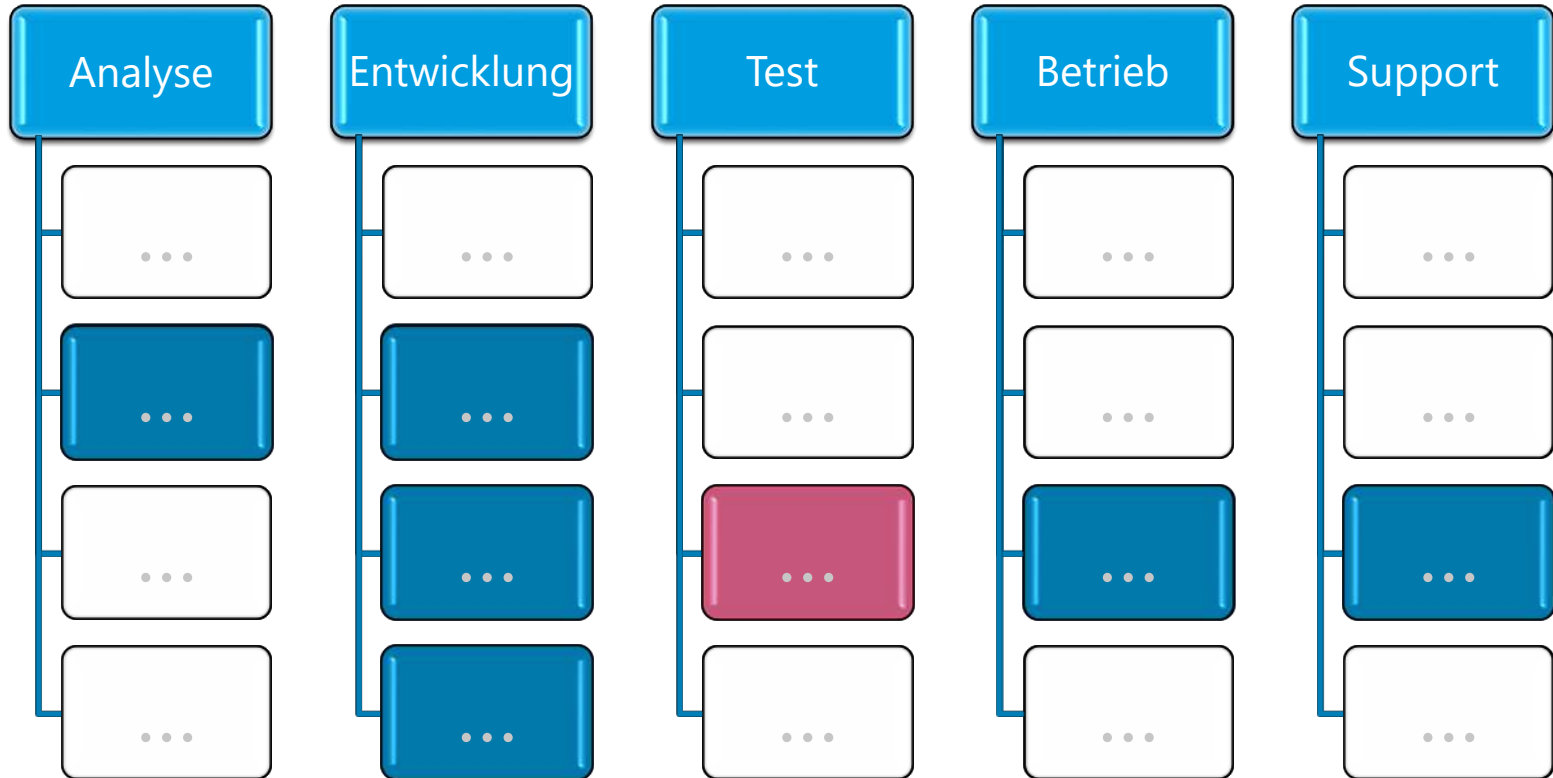
Die neue Kollegin ist da

Modern: Event Storming



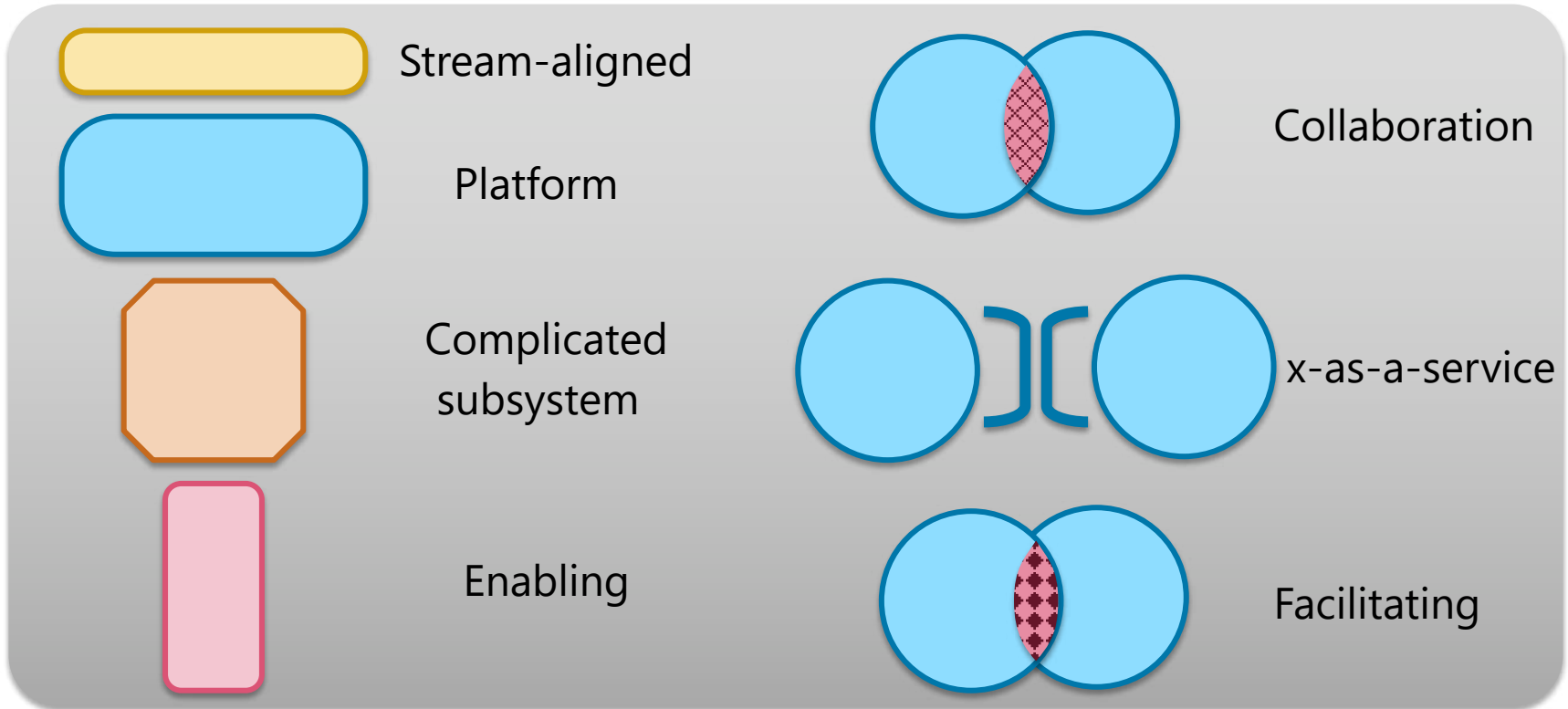
Die neue Kollegin ist da

Klassisch: Projekt kontra Organisation



Die neue Kollegin ist da

Modern: Team Topologies

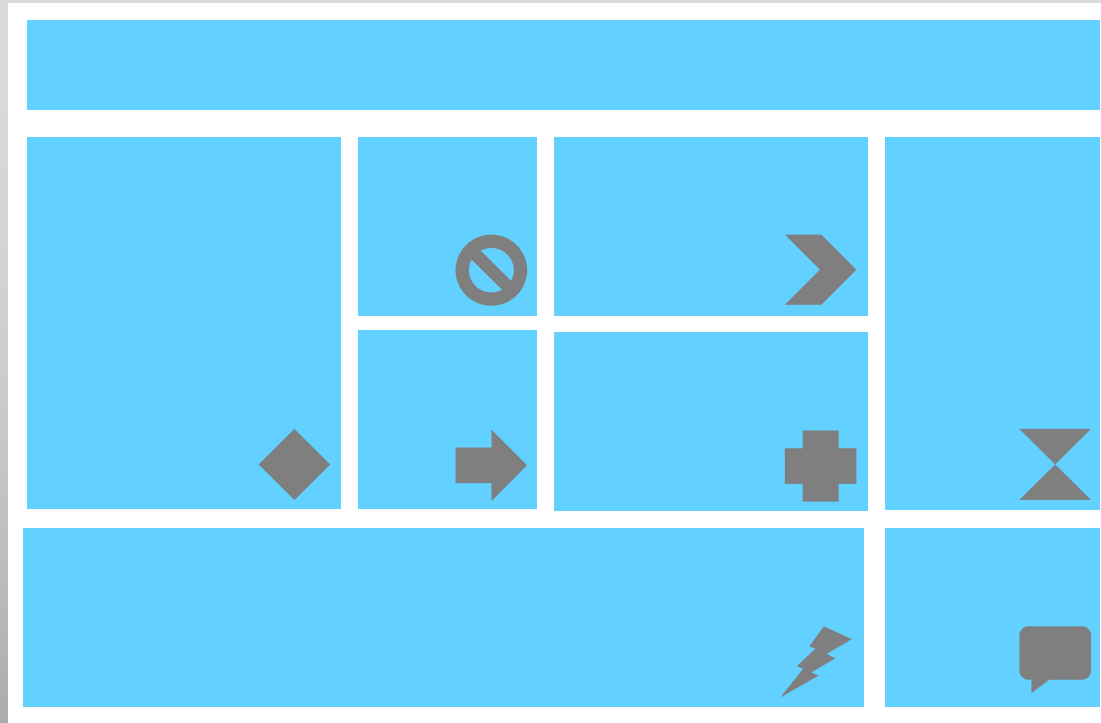


Klassisch: Der „Plan“



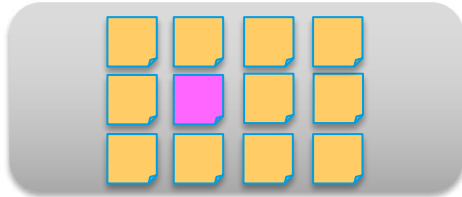
Die neue Kollegin ist da

Modern: Der Canvas

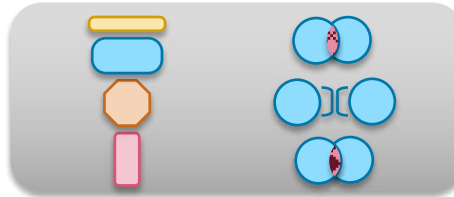


Die neue Kollegin ist da

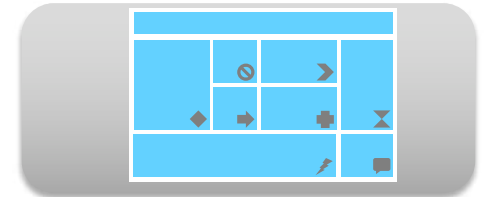
Was waren die Auslöser?



Event Storming



Team Topologies



Canvas

Niemand hat mehr
das gesamte Wissen

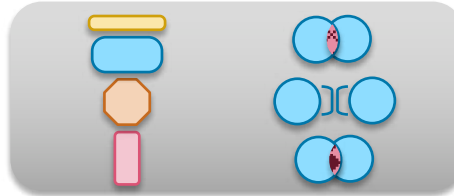
Im Team schneller
„liefern“ können

Selbstständig und
eigenverantwortlich
arbeiten

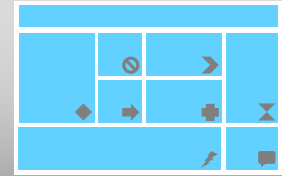
Worauf basieren sie?



Event Storming



Team Topologies



Canvas

Niemand hat mehr
das gesamte Wissen

Im Team schneller
„liefern“ können

Selbstständig und
eigenverantwortlich
arbeiten

Durch einfache
Methoden kann
jeder mitarbeiten

Aufgaben und
Zusammenarbeit
sind klar geregelt

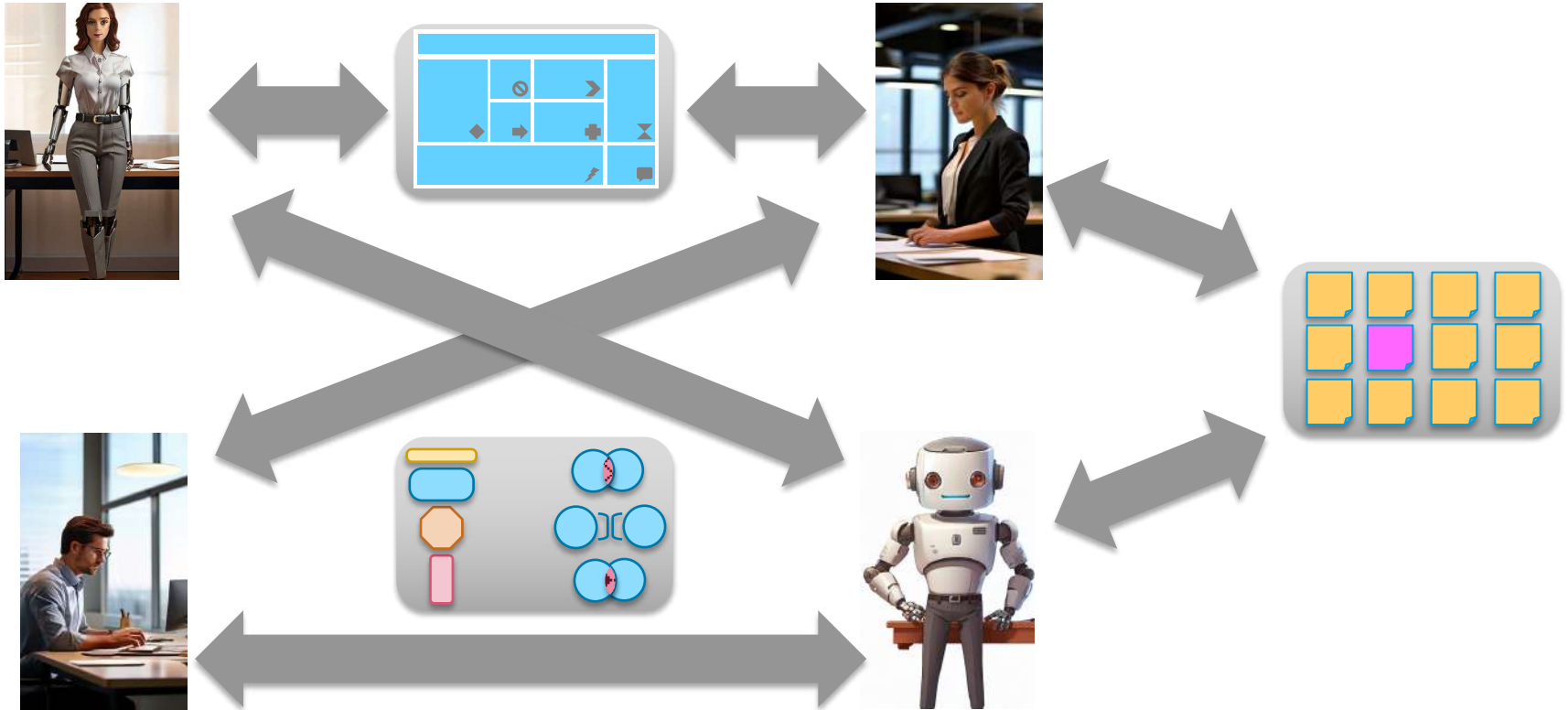
Alles Wesentliche
stets im Blick

Hype oder bleibende Innovation?



Die neue Kollegin ist da

Die Methoden für Menschen helfen auch mit den neuen Kollegen



Die neue Kollegin ist da

Vielen Dank!

www.iks-gmbh.com

› Pascaline:

- ◆ https://commons.wikimedia.org/wiki/File:Pascaline-CnAM_823-1-IMG_1506-white.jpg

› Analytical Mashine

- ◆ [https://de.wikipedia.org/wiki/Analytical_Engine#/media/Datei:Babbages_Analytical_Engine,_1834-1871._\(9660574685\).jpg](https://de.wikipedia.org/wiki/Analytical_Engine#/media/Datei:Babbages_Analytical_Engine,_1834-1871._(9660574685).jpg)
- ◆ https://de.wikipedia.org/wiki/Charles_Babbage#/media/Datei:Engaving_of_Charles_Babbage_from_Mechanics_Magazine.jpg

› Ada Lovelace

- ◆ https://en.wikipedia.org/wiki/Note_G#/media/File:Diagram_for_the_computation_of_Bernoulli_numbers.jpg
- ◆ https://en.wikipedia.org/wiki/Note_G#/media/File:Ada_Lovelace_portrait.jpg

› Zuse

- ◆ https://de.wikipedia.org/wiki/Zuse_Z3#/media/Datei:Elektromagnetischerspeicher_zuse_relais.jpg
- ◆ https://de.wikipedia.org/wiki/Zuse_Z3#/media/Datei:Z3_Deutsches_Museum.JPG

› ENIAC

- ◆ <https://www.flickr.com/photos/epitti/2586212612> (Creative Commons, Attribution 2.0 Generic)
- ◆ https://upload.wikimedia.org/wikipedia/commons/c/ca/AVIDAC_--_First_Argonne_Computer_%281953%29.jpg

> TX-o:

- ◆ https://commons.wikimedia.org/wiki/File:KT3107_transistors.jpg
- ◆ https://miro.medium.com/v2/resize:fit:1400/format:webp/1*O1_WWZn7uiPtDdZrd_i7qQ.jpeg

> Der erste IC

- ◆ https://commons.wikimedia.org/wiki/File:4-%D0%B1%D1%96%D1%82%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80_Intel_4004.jpg
- ◆ https://upload.wikimedia.org/wikipedia/commons/1/18/Busicom_overall_1.jpg

> Die Gewinner sind

- ◆ KI-generiert von DALL-E

> Die Gewinner sind

- ◆ KI-generiert von DALL-E

> Fuzzy-Logic

- ◆ <https://de.wikipedia.org/wiki/Fuzzylogik#/media/Datei:Fuzzy-Operatoren.png>

› Perzeptron

- ◆ <https://upload.wikimedia.org/wikipedia/commons/2/2d/PerceptronOverview.svg>

› Der 1. KI-Winter

- ◆ KI-generiert von neuroflash

› Warum ist Software nur so teuer?

- ◆ KI-generiert von DALL-E

› NLS:

- ◆ [https://de.wikipedia.org/wiki/ON-Line_System#/media/Datei:ON-Line_System_\(NLS\),_SRI_\(1960-1970s\)_-_three-button_mouse_and_chord_keyboard_-_Computer_History_Museum.jpg](https://de.wikipedia.org/wiki/ON-Line_System#/media/Datei:ON-Line_System_(NLS),_SRI_(1960-1970s)_-_three-button_mouse_and_chord_keyboard_-_Computer_History_Museum.jpg)

› Macintosh

- ◆ https://de.wikipedia.org/wiki/Grafische_Benutzeroberfl%C3%A4che#/media/Datei:Mac_Classic.jpg

› Objektorientierung

- ◆ https://commons.wikimedia.org/wiki/File:CPT-OOP-objects_and_classes.svg#/media/File:CPT-OOP-objects_and_classes.svg

› Netzwerke

- ◆ https://commons.wikimedia.org/wiki/File:IP-Paket_Routing_%C3%BCber_Netzwerke.svg

› Unix

- ◆ https://www.flickr.com/photos/rudolf_schuba/153225000/in/photostream/ (Creative Commons, Attribution 2.0 Generic)
- ◆ <https://commons.wikimedia.org/wiki/File:Tux.svg>

› Zu vieles neu:

- ◆ https://commons.wikimedia.org/wiki/File:Server_icon_CC0.svg
- ◆ <https://commons.wikimedia.org/wiki/File:Database-icon.svg>
- ◆ <https://www.vectorstock.com/royalty-free-vector/outline-object-oriented-programming-icon-isolated-vector-28254018>
- ◆ https://commons.wikimedia.org/wiki/File:Noun_project_network_icon_1365244_cc.svg
- ◆ https://commons.wikimedia.org/wiki/File:Brain_with_a_head_outline_and_grey_and_blue_puzzle_pieces.jpg

› 2. Softwarekrise

- ◆ KI-generiert von DALL-E

Abbildungsverzeichnis

> 2. KI-Winter

- ◆ KI-generiert von DALL-E

> Timeline

- ◆ <https://vectorportal.com/de/vector/winter-landschaft-vektor-bild/27130>
- ◆ <https://commons.wikimedia.org/wiki/File:Database-icon.svg>

> Deep Blue

- ◆ https://de.wikipedia.org/wiki/Deep_Blue#/media/Datei:Deep_Blue.jpg
- ◆ <https://commons.wikimedia.org/wiki/File:Kasparov-23.jpg>

> Komplexität

- ◆ KI-generiert von DALL-E

> World Wide Web

- ◆ KI-generiert von DALL-E

› Der moderne PC

- ◆ <https://www.pexels.com/de-de/foto/schreibtisch-buro-tisch-technologie-7238759>
- ◆ <https://www.flickr.com/photos/e2/393217813>

› Das Smartphone

- ◆ <https://www.trustedreviews.com/wp-content/uploads/sites/54/2023/11/Best-smartphone-3-920x518.jpg>
- ◆ https://commons.wikimedia.org/wiki/File:Apple_Devices.jpg

› Multicore

- ◆ <https://www.flickr.com/photos/3336/27830149309> (Creative Commons, Attribution 2.0 Generic)
- ◆ https://commons.wikimedia.org/wiki/File:Intel_CPU_Core_i7_6700K_Skylake_top.jpg

› Rechenzentren

- ◆ KI-generiert von DALL-E

› Quantencomputer

- ◆ KI-generiert von DALL-E

Abbildungsverzeichnis

› BigData

- ◆ <https://commons.wikimedia.org/wiki/File:Internet-icon.png>
- ◆ <https://commons.wikimedia.org/wiki/File:Mobile-phone.png>
- ◆ https://commons.wikimedia.org/wiki/File:Cloud_computing_icon.svg

› Timeline:

- ◆ <https://commons.wikimedia.org/wiki/File:Docker-svgrepo-com.svg>
- ◆ <https://picryl.com/media/darpa-big-data-a1d68a>
- ◆ <https://bostondynamics.com/blog/electric-new-era-for-atlas/>
- ◆ https://commons.wikimedia.org/wiki/File:Big_data_%2821036%29_-_The_Noun_Project.svg

› AlphaGo

- ◆ [https://upload.wikimedia.org/wikipedia/commons/3/38/AlphaGo_Fan_Huiren_aurka_\(2\).png](https://upload.wikimedia.org/wikipedia/commons/3/38/AlphaGo_Fan_Huiren_aurka_(2).png)
- ◆ <https://citaty-slavyh.sk/media/authors/lee-sedol.600w.webp>

› Die neue Kollegin ist da

- ◆ KI-generiert von DALL-E

› KI - Werkzeug oder Kollegin

- ◆ Schraubenschlüssel: Foto Jan Radeck
- ◆ Generiert mit Playground.ai: "office robot standing at an office desk wearing blouse and trousers flat chest small breasts"

› Die neue Kollegin steht für ein Team neuer Kolleginnen

- ◆ Generiert mit Playground.ai: "a robot assistant," - Ergebnis A
- ◆ Generiert mit Playground.ai: "a robot assistant," - Ergebnis B
- ◆ Generiert mit Playground.ai: "a robot assistant in an office"
- ◆ Generiert mit Playground.ai: "robot assistant standing at an office desk"

› Klassisch: Analyse

- ◆ Generiert mit Playground.ai: "an archaeologist in a cave excavating a vase panorama"

› Klassisch: Der Plan

- ◆ Generiert mit Playground.ai: "thick stack of loose pages covered in dust on the desk of a dimly lit room"

› Hype oder bleibende Innovation:

- ◆ Generiert mit Playground.ai: "robot standing at an office desk wearing trousers flat chest ballerina shoes"
- ◆ Generiert mit Playground.ai: "female office worker standing at desk"
- ◆ Generiert mit Playground.ai: "office worker standing at desk"
- ◆ Generiert mit Playground.ai: "question mark"