Tech Talks

4GL Dreams

Jeremy Taylor, Head of Product, XTDB

2025-12-05

The Recurring Dream of Programming Without Programming

BE DECLARATIVE

Describe WHAT you want.

Let the machine search for HOW.

BUT

verifying output must be cheap!

"Men will set the goals, formulate the hypotheses, determine the criteria, and perform the evaluations. Computing machines will do the routinizable work."

J.C.R. Licklider, 1960

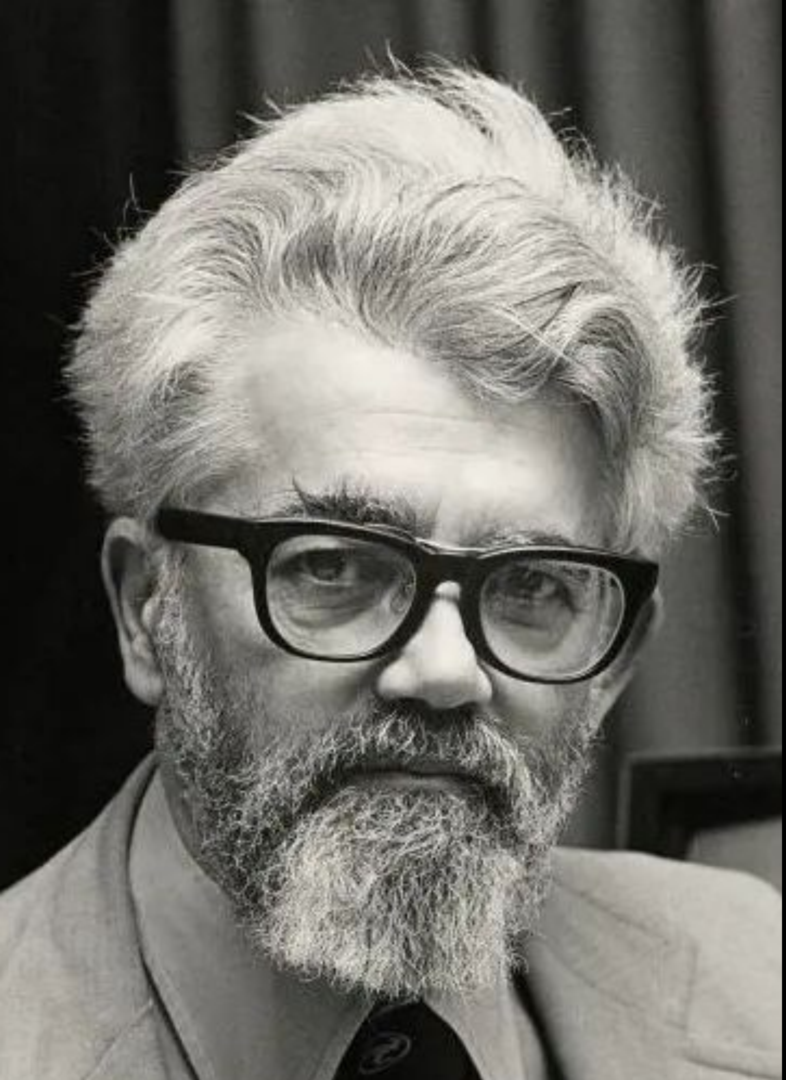
1. DECLARE what you want
2. System SEARCHES for how
3. Human VERIFIES the output

“Nobody believed that I had a running compiler“

Grace Hopper's
A-0 Compiler (1952)

“Can Programming Be Liberated from the von Neumann Style?”

John Backus,
FORTRAN (1957),
Turing Award Lecture (1977)



“Lisp owes its survival specifically to the fact that its programs are lists, which everyone, including me, has regarded as a disadvantage.”

John McCarthy
LISP (1958),
Early History of LISP (1979)

And then came...

COBOL (1959)

```
MULTIPLY HOURS BY RATE
GIVING WAGES
```

DECLARE: business logic in English-like syntax
SEARCH: compiler handles details
VERIFY: can a manager read it? (supposedly)

A Relational Model of Data for Large Shared Data Banks (1970)

+

SEQUEL: A structured English query language (1974)

SQL: The Greatest 4GL So Far

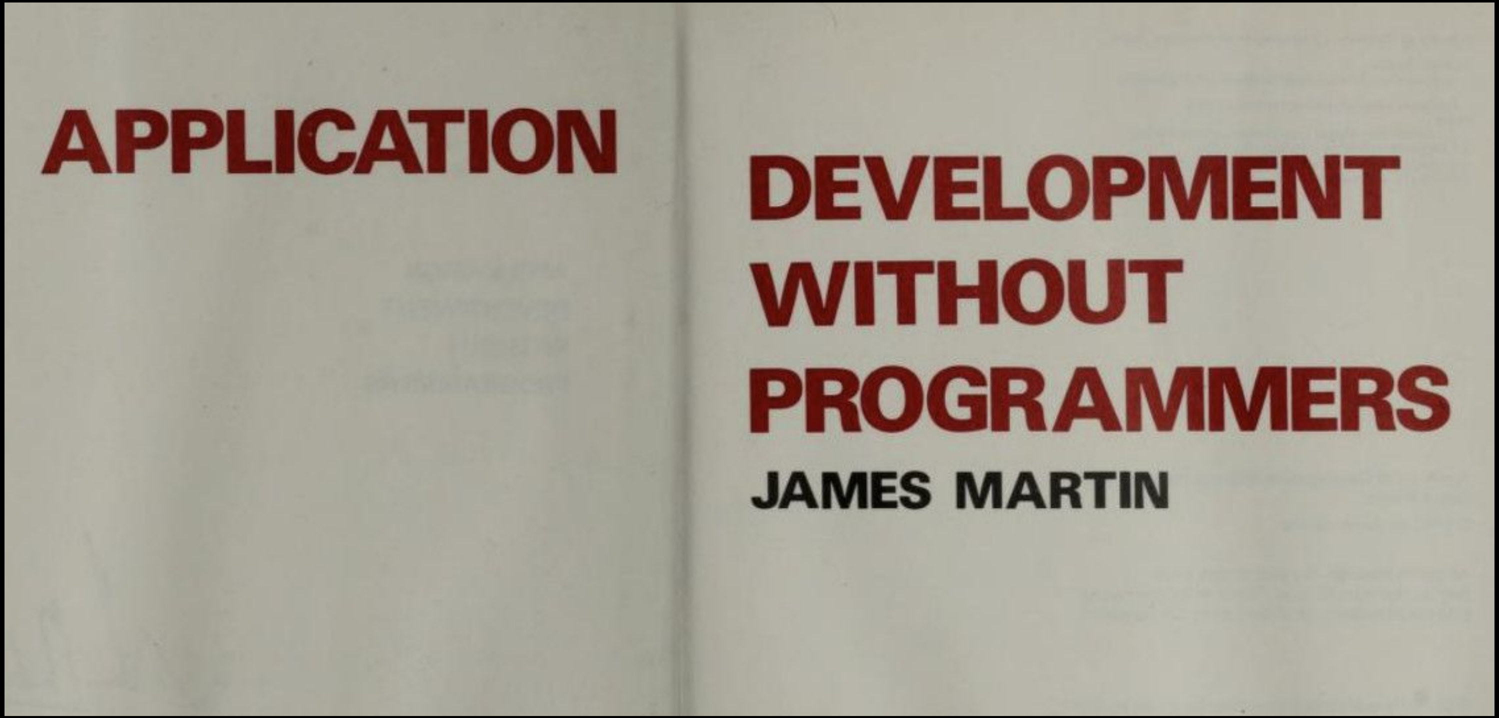
```
SELECT name FROM employees
WHERE department = 'Sales'
```

DECLARE: describe the data you want
SEARCH: query optimizer finds execution plan
VERIFY: inspect the result set

Prolog (1972): Also Ran

```
mortal(X) :- human(X).
human(socrates).
?- mortal(socrates).  → yes
```

DECLARE: logical facts and rules
SEARCH: unification + backtracking
VERIFY: is the answer logically correct?

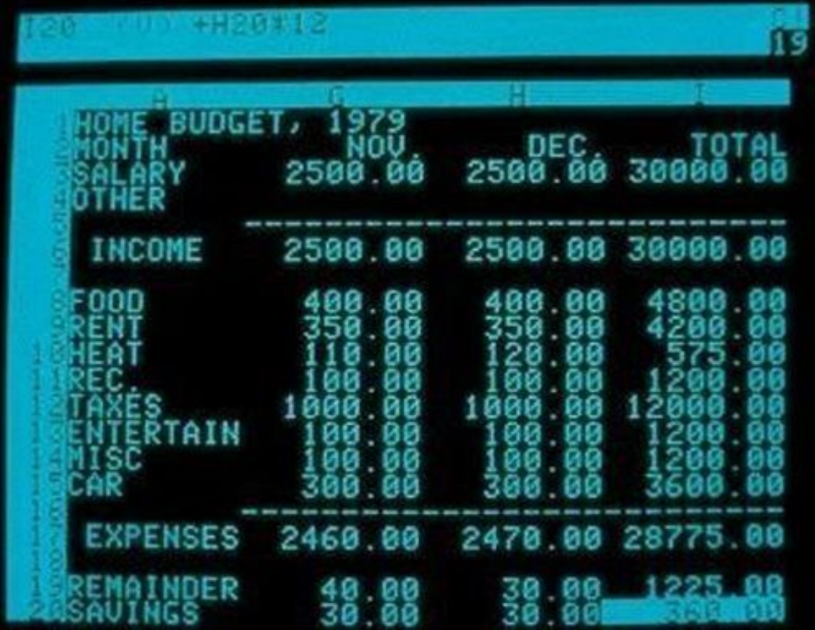


APPLICATION DEVELOPMENT WITHOUT PROGRAMMERS

JAMES MARTIN

"4GL" (1981)

...spreadsheets! VisiCalc (1979)



```
I20 (V) +H20*12                                        C!
                                                       19
          A            G           H           I
 1 HOME BUDGET, 1979
 2 MONTH             NOV.        DEC.      TOTAL
 3 SALARY          2500.00     2500.00   30000.00
 4 OTHER
 5               ----------------------------------
 6  INCOME         2500.00     2500.00   30000.00
 7
 8 FOOD             400.00      400.00    4800.00
 9 RENT             350.00      350.00    4200.00
10 HEAT             110.00      120.00     575.00
11 REC.             100.00      100.00    1200.00
12 TAXES           1000.00     1000.00   12000.00
13 ENTERTAIN        100.00      100.00    1200.00
14 MISC             100.00      100.00    1200.00
15 CAR              300.00      300.00    3600.00
16               ----------------------------------
17  EXPENSES       2460.00     2470.00   28775.00
18
19 REMAINDER         40.00       30.00    1225.00
20 SAVINGS           30.00       30.00     360.00
```

DECLARE: formulas describing relationships
SEARCH: dependency graph evaluation
VERIFY: you can see the numbers right there :)

What’s the problem?

The nonprogrammer still has to think like a programmer *(paraphrasing)*

Fred Brooks, "No Silver Bullet" (1986)

Solutions?

Andrej Karpathy's "Software 2.0"

"Software 1.0 is the computer code you write; Software 2.0 is essentially the weights of neural networks. You don't write it directly but create these parameters by adjusting the dataset and running the optimizer." - ***Software Is Changing (Again)*** (2025)

DECLARE: examples of desired behavior (dataset)
SEARCH: gradient descent over weight space
VERIFY: test set accuracy

Andrej Karpathy's "Software 3.0"

"Neural networks have become programmable through large language models [...] **The hottest new programming language is English**"

DECLARE: natural language prompt
SEARCH: autoregressive sampling
VERIFY: ???

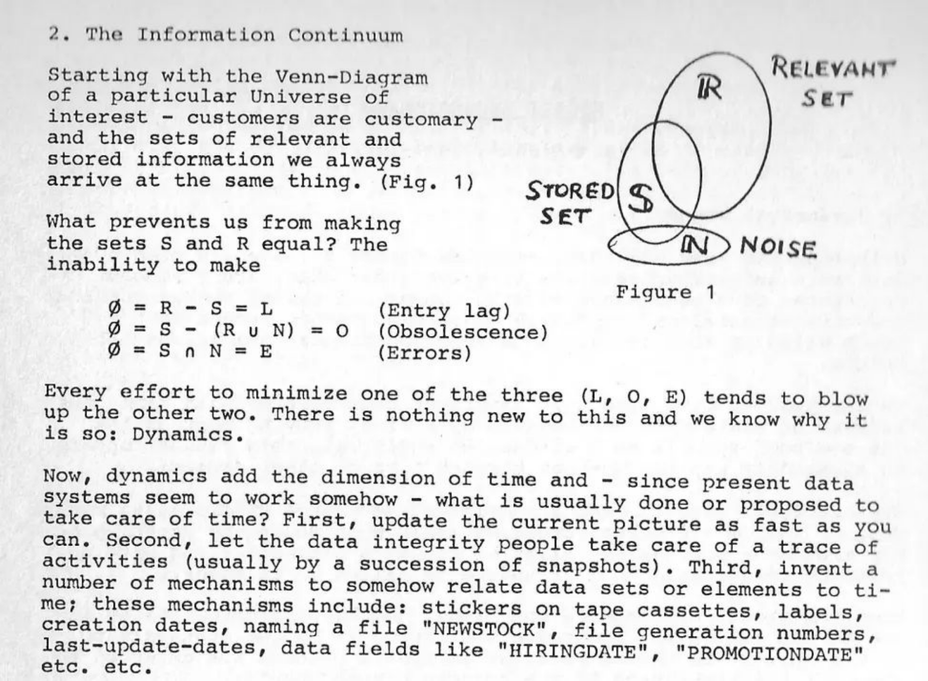
Verification is Key

“The more a task is verifiable, the more amenable it is to automation. Tasks that are verifiable progress rapidly (math, code), while others lag (creative, strategic)." - Karpathy

And The Dream Goes All The Way **Down**

DECLARE: algorithm / blocked structure
SEARCH: compiler explores implementation space
VERIFY: benchmark performance *(measurable!)*

Databases and Schueler's Vision



2. The Information Continuum

Starting with the Venn-Diagram of a particular Universe of interest - customers are customary - and the sets of relevant and stored information we always arrive at the same thing. (Fig. 1)

RELEVANT SET
R
STORED SET
S
N
NOISE

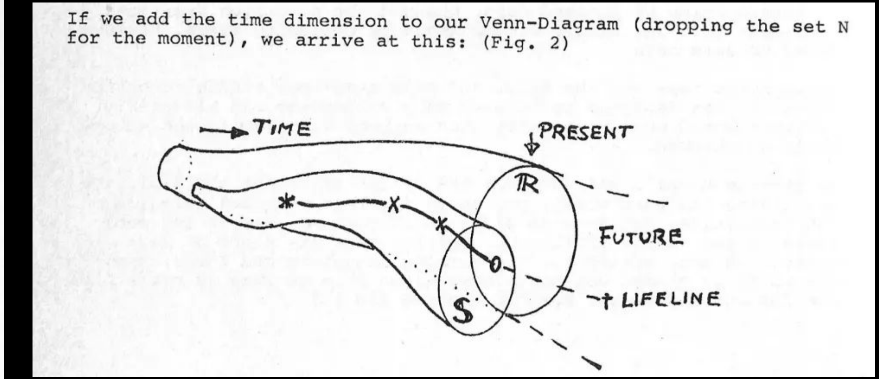
Figure 1

What prevents us from making the sets S and R equal? The inability to make

$\emptyset = R - S = L$ (Entry lag)
$\emptyset = S - (R \cup N) = O$ (Obsolescence)
$\emptyset = S \cap N = E$ (Errors)

Every effort to minimize one of the three (L, O, E) tends to blow up the other two. There is nothing new to this and we know why it is so: Dynamics.

Now, dynamics add the dimension of time and - since present data systems seem to work somehow - what is usually done or proposed to take care of time? First, update the current picture as fast as you can. Second, let the data integrity people take care of a trace of activities (usually by a succession of snapshots). Third, invent a number of mechanisms to somehow relate data sets or elements to time; these mechanisms include: stickers on tape cassettes, labels, creation dates, naming a file "NEWSTOCK", file generation numbers, last-update-dates, data fields like "HIRINGDATE", "PROMOTIONDATE" etc. etc.



If we add the time dimension to our Venn-Diagram (dropping the set N for the moment), we arrive at this: (Fig. 2)

UPDATE RECONSIDERED, Ben-Michael Schueler (1977)

Neurosymbolic Architectures

"Men will set the goals, formulate the hypotheses, determine the criteria, and perform the evaluations. Computing machines will do the routinizable work."

J.C.R. Licklider, 1960

The End

@refset