Growing a Language

Joe Armstrong

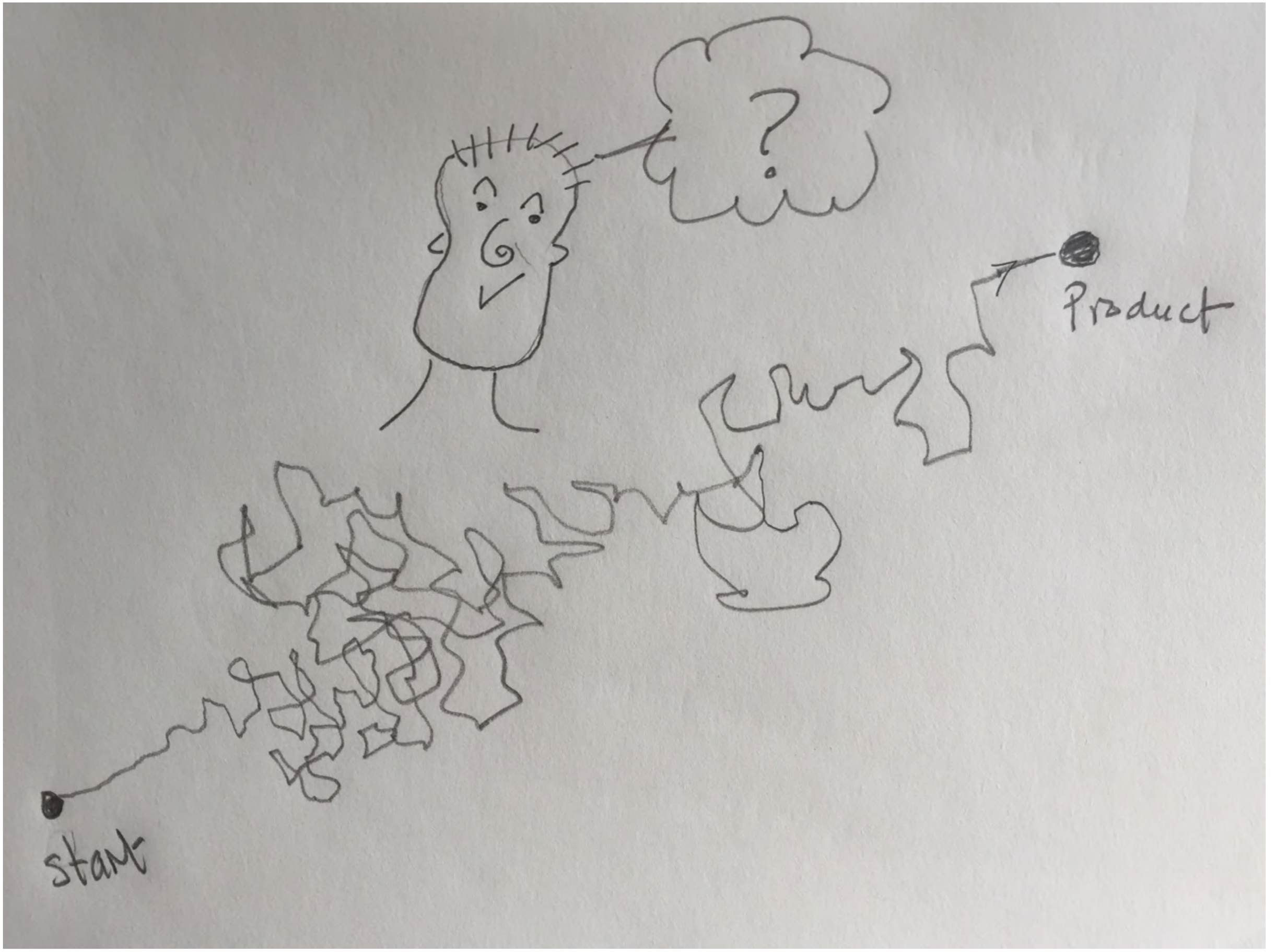
- Make a prototype
- Get some users
- Write some documentation
- Make a “proper” version
- Make a course
- Make a product
- Write a book
- Sell the company
- Grow a community
- Fight technical battles
- Give talks
- Go to meetings
- Write code
- Talk to financiers

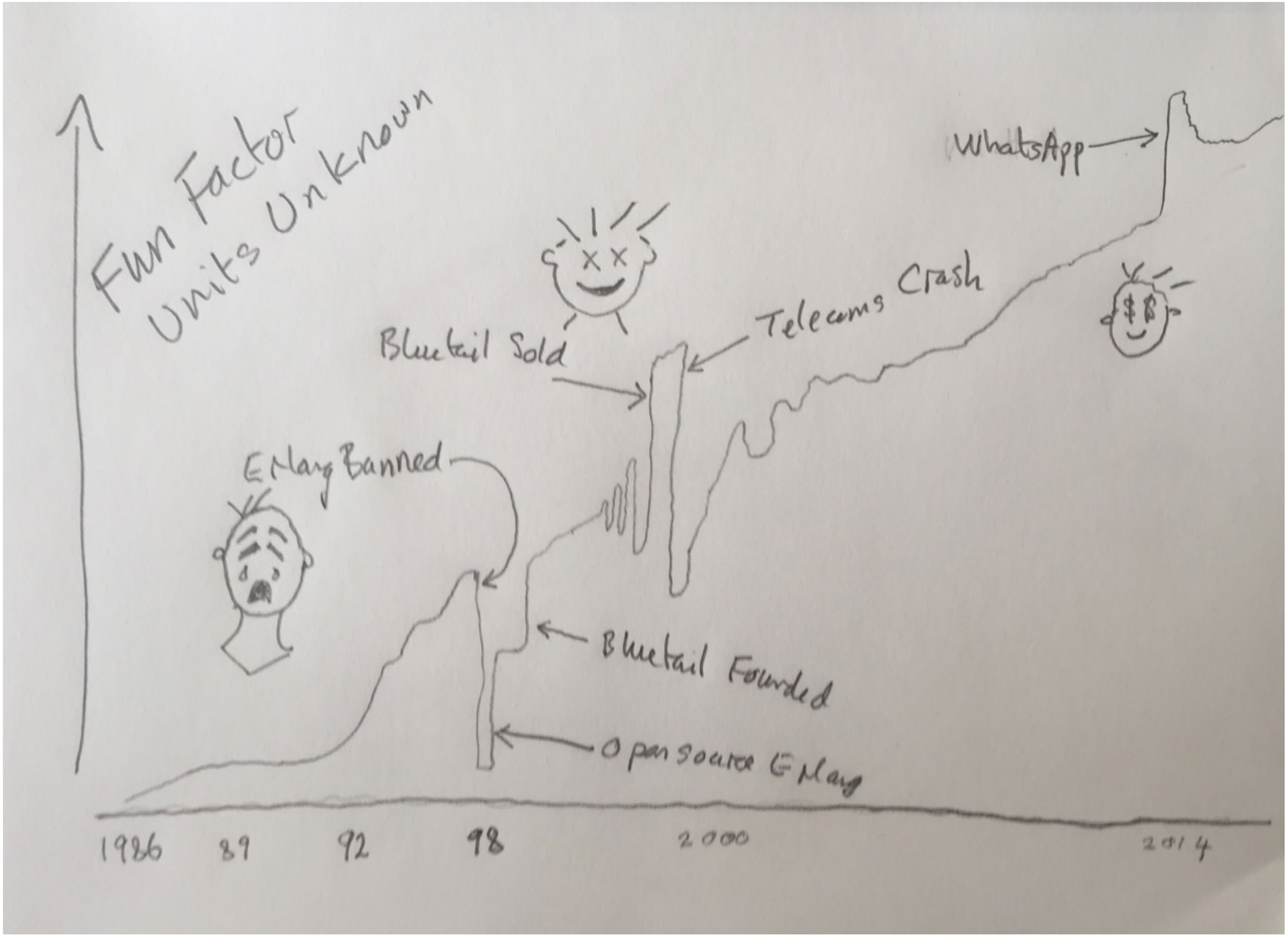
Product Development

- Specify
- Build
- Test
- Sell
- $$$

Agile

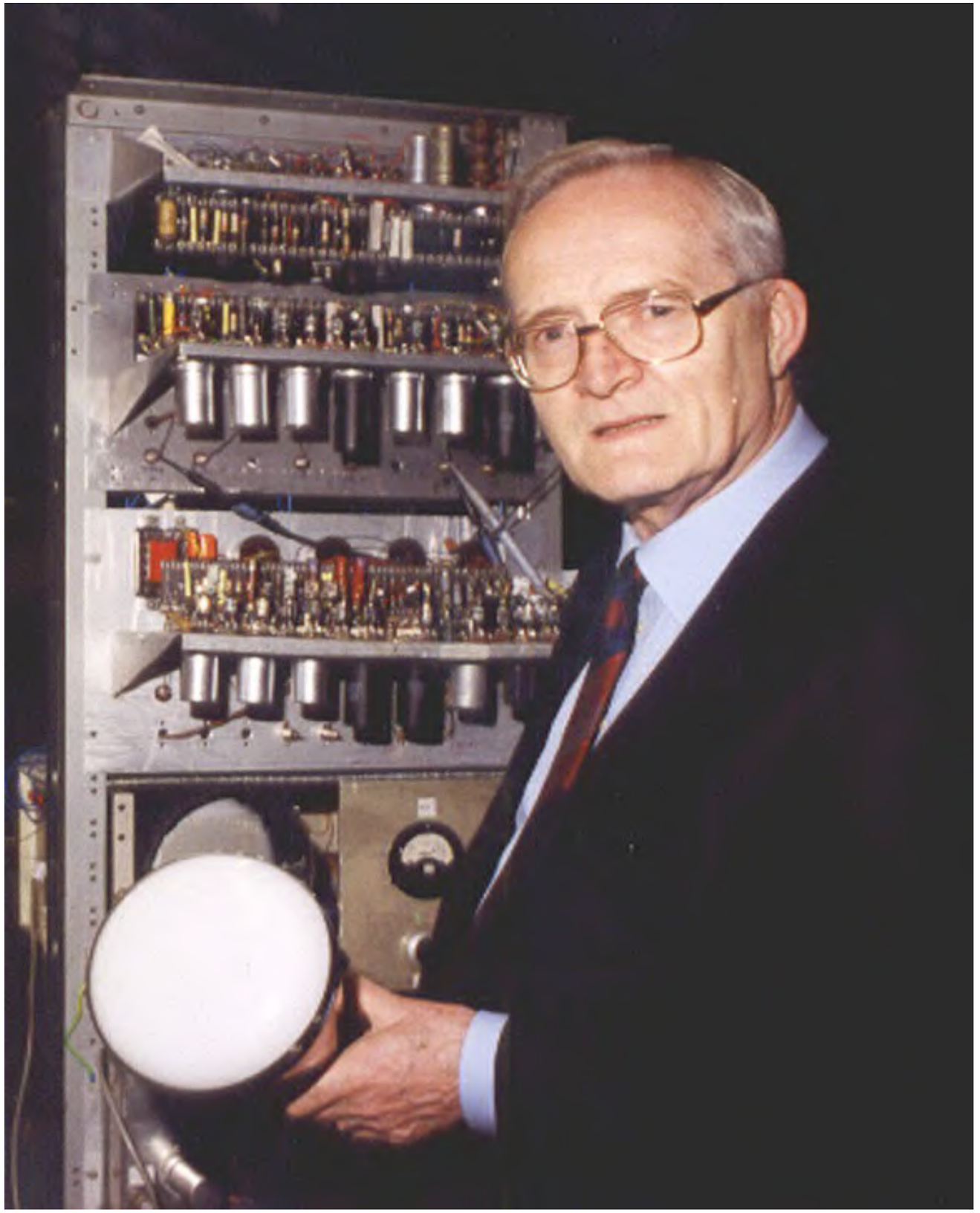
Scrum


?
Product
start


Fun Factor
Units Unknown
WhatsApp
Telecoms Crash
Bluetail Sold
EMarg Banned
Bluetail Founded
Open source EMarg
1986
89
92
98
2000
2014

History

21 June 1948



19/7/49 — Kilburn Highest Factor Routine (amended) —

Instn.	C	25	26	27	Line	0 1 2 3 4 5	13 14 15
-24 to C	$-G_1$	-	-	-	1	0 0 0 1 1	0 1 0
+ to 26			$-G_1$		2	0 1 0 1 1	1 1 0
-26 to C	G_1				3	0 1 0 1 1	0 1 0
+ to 27			$-G_1$	G_1	4	1 1 0 1 1	1 1 0
-23 to C	a	T_{n-1}	$-G_n$	G_n	5	1 1 1 0 1	0 1 0
Sub 27	$a-nG_1$				6	1 1 0 1 1	0 0 1
Test					7	—	0 1 1
Add 20 to C					8	0 0 1 0 1	1 0 0
Sub. 26	T_n				9	0 1 0 1 1	0 0 1
+ to 25		T_n			10	1 0 0 1 1	1 1 0
-26 to C					11	1 0 0 1 1	0 1 0
Test					12	—	0 1 1
Stop	0	0	$-G_N$	G_N	13		1 1 1
-26 to C	G_n	T_n	$-G_n$	G_n	14	0 1 0 1 1	0 1 0
Sub. 27	G_{n+1}				15	1 0 1 0 1	0 0 1
+ to 27	G_{n+1}			G_{n+1}	16	1 1 0 1 1	1 1 0
-27 to C	$-G_{n+1}$				17	1 1 0 1 1	0 1 0
+ to 26			$-G_{n+1}$		18	0 1 0 1 1	1 1 0
22 to 26		T_n	$-G_{n+1}$	G_{n+1}	19	0 1 1 0 1	0 0 0

20	-3	1 0 1 1 1 to C
21	1	1 0 0 0 0
22	4	0 0 1 0 0

23	-a	
24	G_1	

	Instr.	Final
25	—	T_N (= 0)
26	—	$-G_N$
27	—	G_N

or 1 0 1 0 0

+ 0 0 0

'Nothing was ever the same again.'

Freddie Williams, computer pioneer, 1976

Nothing was ever
the same again

Context

Languages that I could learn vs. time

1948	1977	1986	2015	2053
1	3	25	2500	Too many

Hewlett Packard Vectra 45945A

Introduced 1985

- 256 KB RAM - Expandable to 3.64 MB with expansion slots
- 80286 CPU at 8Mz
- 20MB or 40MB hard disk
- 2 x 5.25 inch floppy drives

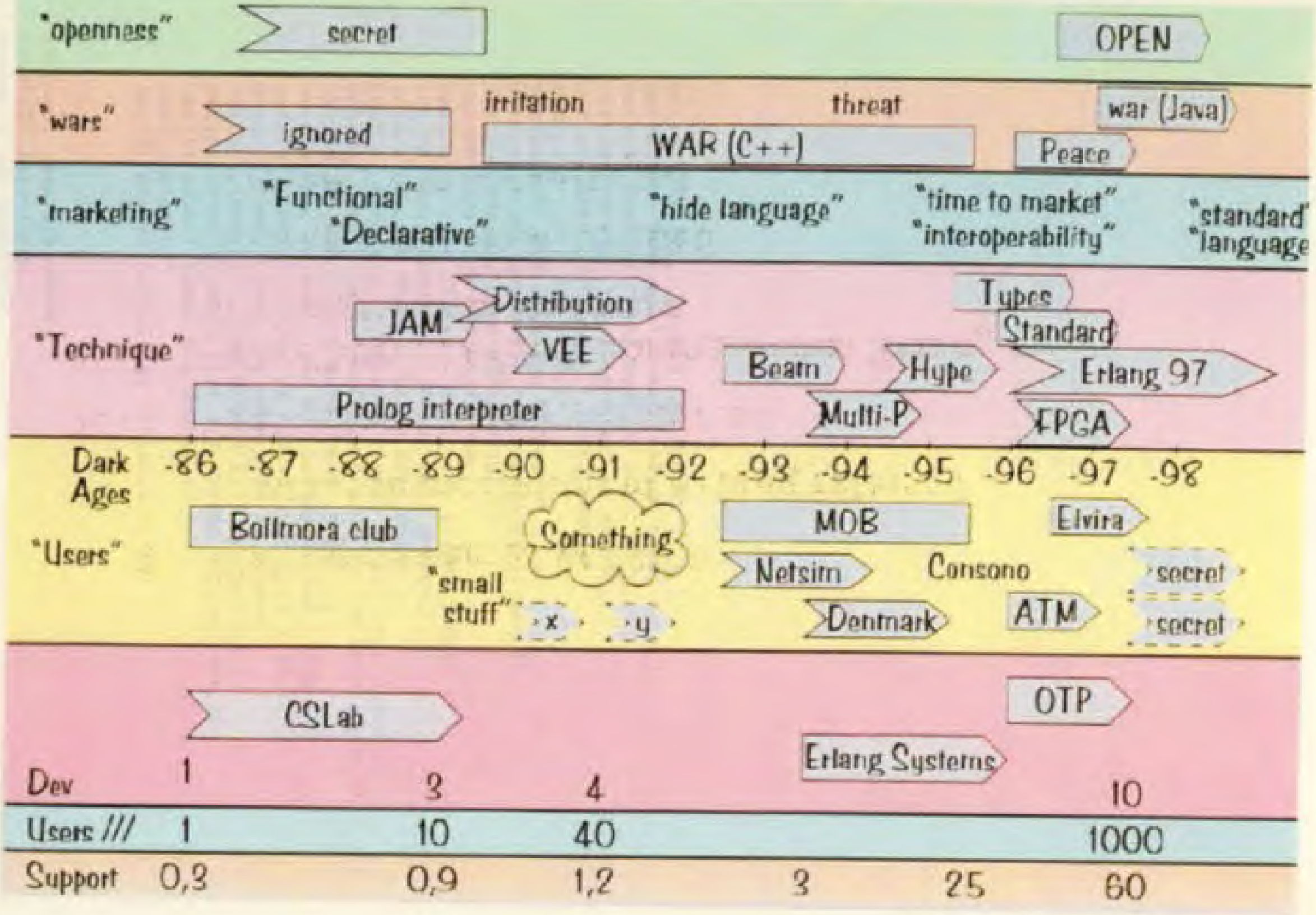
Source: http://www.hpmuseum.net/

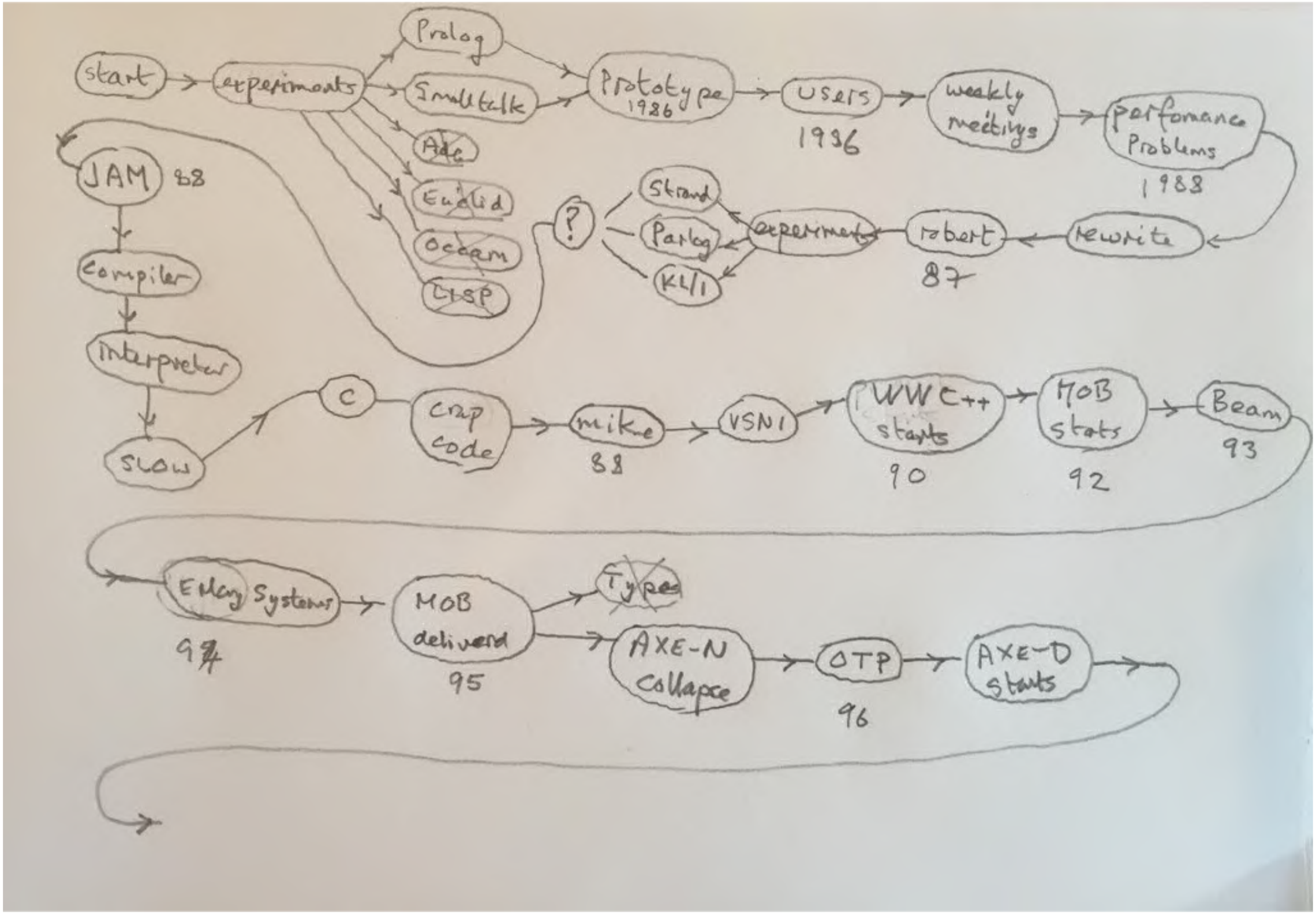
2015

- 8 Gbytes memory (x 32,000)
- 4 – core – 2.5GHz (x 1,000)
- 250 GB SSD (x10,000)

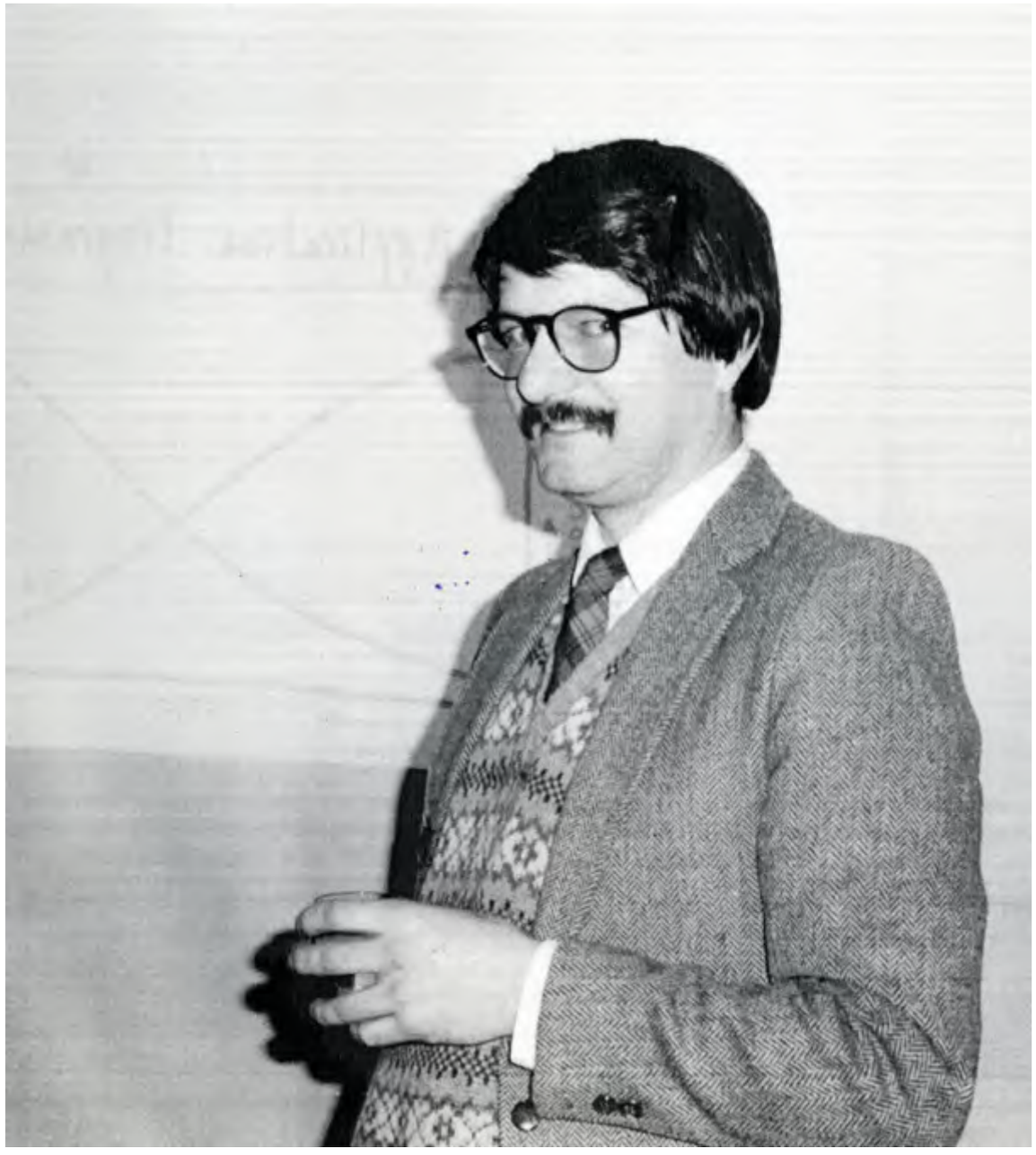
The machine is 1000x faster – so a 60 second boot time in 1985 should be 60ms in 2014 :-)

What went wrong?


"openness"
secret
OPEN
"wars"
ignored
irritation
threat
WAR (C++)
war (Java)
Peace
"marketing"
"Functional"
"Declarative"
"hide language"
"time to market"
"interoperability"
"standard"
"language
"Technique"
JAM
Distribution
VEE
Prolog interpreter
Beam
Hype
Multi-P
Types
Standard
Erlang 97
FPGA
Dark Ages
-86
-87
-88
-89
-90
-91
-92
-93
-94
-95
-96
-97
-98
"Users"
Boilmora club
Something
"small stuff"
x
y
MOB
Netsim
Denmark
Consono
ATM
Elvira
secret
secret
CSLab
OTP
Erlang Systems
Dev
1
3
4
10
Users ///
1
10
40
1000
Support
0,3
0,9
1,2
3
25
60

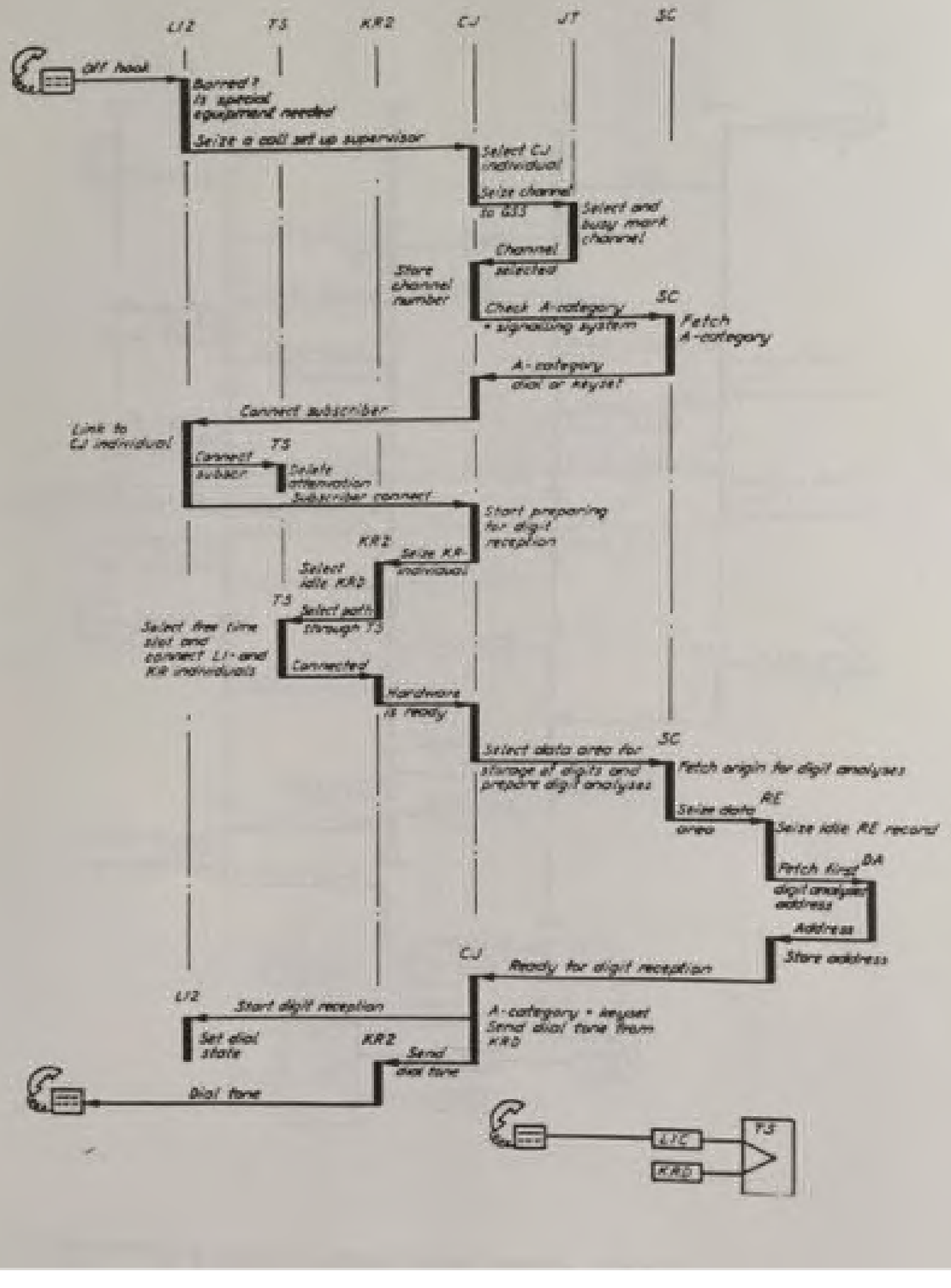

start
experiments
Prolog
Smalltalk
Ada
Euclid
Occam
LISP
Prototype
1986
Users
1986
weakly meetings
Performance Problems
1988
rewrite
robert
87
experiments
Strand
Parlog
KL/1
?
JAM
88
compiler
interpreter
SLOW
C
crap code
mike
88
VSN1
WWC++
starts
90
MOB
stats
92
Beam
93
EMay Systems
97
MOB
delivered
95
Types
AXE-N
Collapse
OTP
96
AXE-D
starts

1986



Start with a problem

- How can we program telephony? - Erlang
- How can we describe a page? - Postscript
- How can we program a set-top box? - Java
- How can we experiment with types? - Haskell
- How can we parse natural language? - Prolog
- How can we do statistical computations? - R
- ...
- How can we confuse people? - Xcode


LI2
TS
KR2
CJ
JT
SC
Off hook
Barred? Is special equipment needed
Seize a call set up supervisor
Select CJ individual
Seize channel to SIS
Select and busy mark channel
Channel selected
Store channel number
Check A-category + signalling system
SC
Fetch A-category
A-category dial or keyset
Connect subscriber
Link to CJ individual
Connect subscr
TS
Delete attenuation
Subscriber connect
Start preparing for digit reception
KR2
Seize KR individual
Select idle KR2
TS
Select path through TS
Select free time slot and connect LI- and KR individuals
Connected
Hardware is ready
Select data area for storage of digits and prepare digit analysis
SC
Fetch origin for digit analysis
Seize data area
RE
Seize idle RE record
Fetch first digit analysis address
DA
Address
Store address
CJ
Ready for digit reception
LI2
Start digit reception
A-category + keyset Send dial tone from KRD
Set dial state
KR2
Send dial tone
Dial tone
LTC
KRD
TS

The Reduction machine

P → a Q R b
Q → R c
R → d

P
a Q R b
Q R b
R c R b
d c R b
c R b
R b
d b
b
ε

P,Q,R, ... are functions
a,b,c … are primitives

4 reduction machines running in parallel

P
a Q R b
Q R b
R c R b
d c R b
c R b
R b
d b
b
ε

P
a Q R b
Q R b
R c R b
d c R b
c R b
R b
d b
b
ε

P
a Q R b
Q R b
R c R b
d c R b
c R b
R b
d b
b
ε

P
a Q R b
Q R b
R c R b
d c R b
c R b
R b
d b
b
ε

Suspending and resuming a computation

P
a Q R b
Q R b
R c R b
d c R b

Store the last row in a database

Suspend

Fetch Computation from database

Resume

d c R b
c R b
R b
d b
b
ε

Message Passing

P → Q receive R S
Q → a

P1 = spawn(P)
Q receive R S
a receive R S
receive R S

Suspends

x U R S

T → K send P1 {x U} S
...
...

...
send P1 {x U} S
S

The story so far ...

- A suspend/resume mechanism
- A message massing mechanism
- A reduction mechanism
- A process spawn mechanism

Needs one more thing

Links

If a process crashes a message is sent to the *linked* process

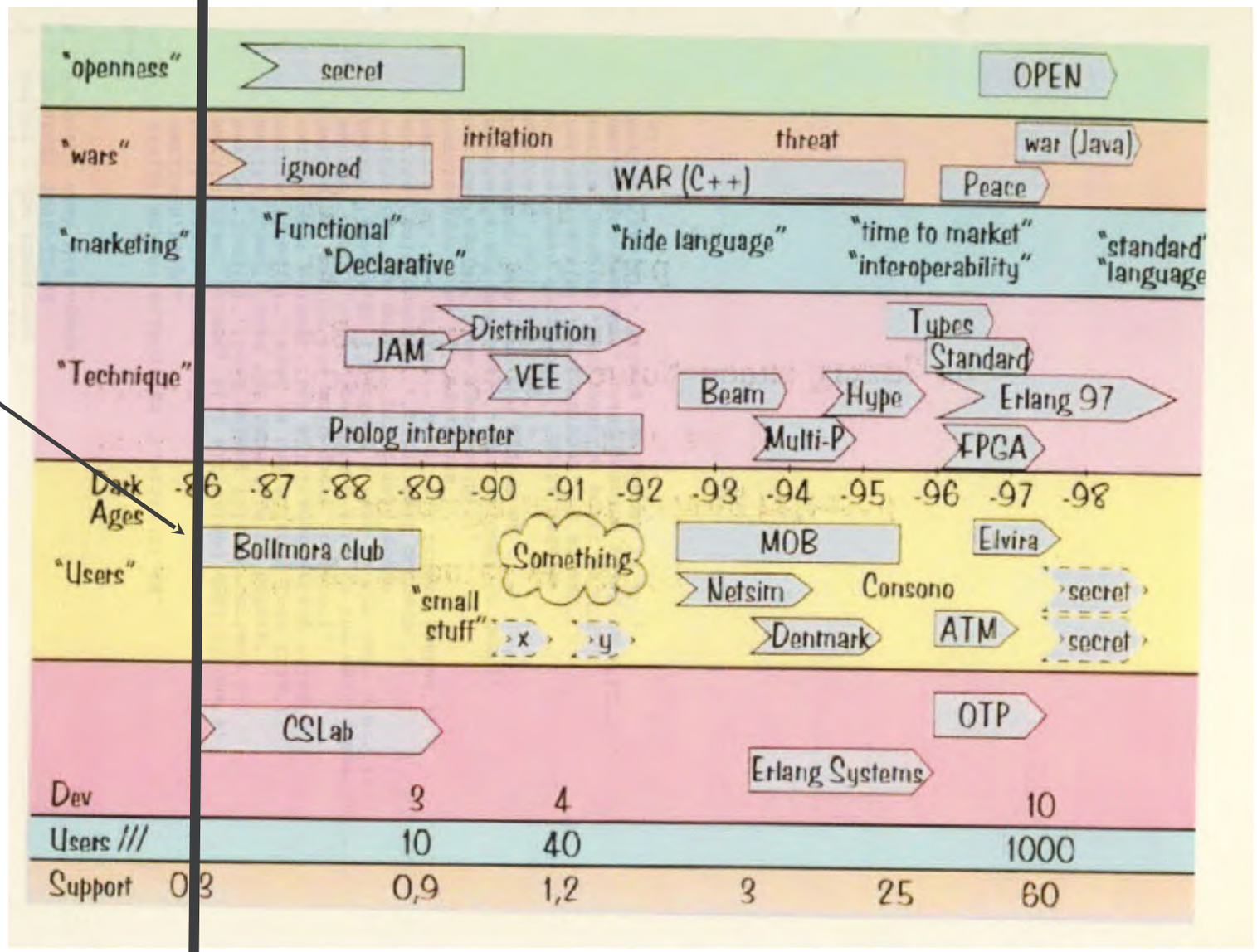
```
/*
 * $HOME/erlang.pro
 *
 *                Copyright (c) 1988 Ericsson Telecom
 *
 * Author: Joe Armstrong
 * Creation Date: 1988-03-24
 * Purpose:
 *      main reduction engine
 *
 * Revision History:
 *      88-03-24        Started work on multi processor version
 *                      of erlang
 *      88-03-28        First version completed (Without timeouts)
 *      88-03-29        Correct small errors
 *      88-03-29        Changed 'receive' to make it return the pair
 *                      msg(From,Mess)
 *      88-03-29        Generate error message when out of goals
 *                      i.e. program doesn't end with terminate
 *      88-03-29        added trace(on), trace(off) facilities
 *      88-03-29        Removed Var := {....} , this can be achieved
 *                      with {..}
 *      88-05-27        Changed name of file to erlang.pro
 *                      First major revision started - main changes
 *                      Complete change from process to channel
 *                      based communication
 *                      here we (virtually) throw away all the
 *                      old stuff and make a bloody great data base
 *      88-05-31        The above statements were incorrect much better
 *                      to go back to the PROPER way of doing things
 *                      long live difference lists
 *      88-06-02        Reds on run([et5]) = 245
 *                      changing the representation to separate the
 *                      environment and the process - should improve things
 *                      It did .... reds = 283 - and the program is nicer!
 *      88-06-08        All pipe stuff working (pipes.pro)
 *                      added code so that undefined functions can return
 *                      values
```

4 days for a rewrite

Not so fast

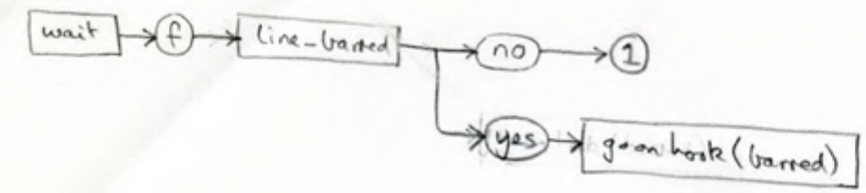
We have a language
all we now need are some users
and an application

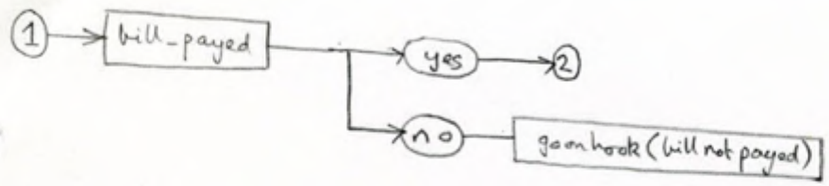
KERSTIN ÖDLING
Ericsson Business Networks AB

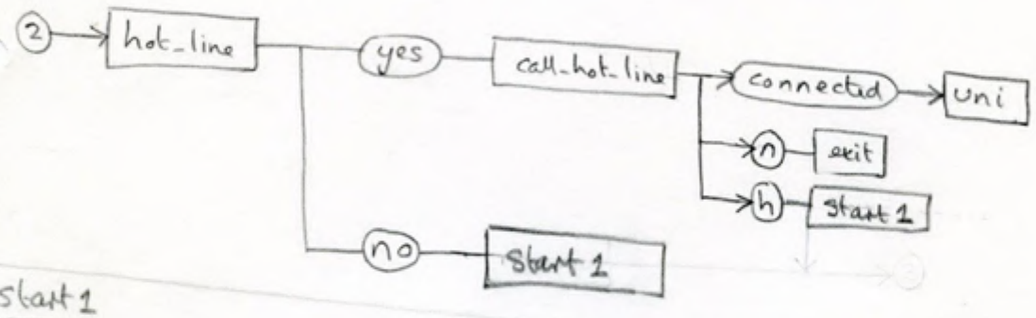


MD110

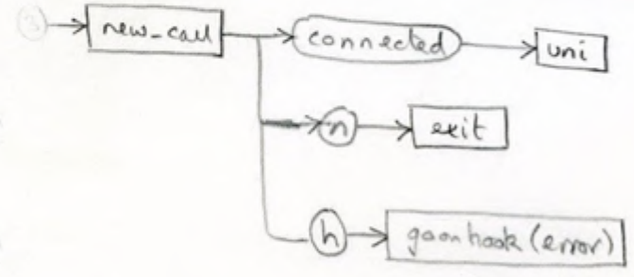
start (start it off)







start 1



Notes

1) an exit from [call_hot_line] of (h) allows the user to attempt a new call

case F +

```
function uni returns none.

1 # uni --->
        case(wait,[
            n => [term,exit],
            h => [hold,
                  case(new_call,[
                    connected => multi,
                    n => case(gone_away,[
                      yes => exit,
                      no => [conv,uni]
                      ]),
                    h => [conv,uni]
                    ])
                ]
        ]).
```

1988

The prototype works let's make a product

- Courses
- Documentation
- Performance
- Support

Courses

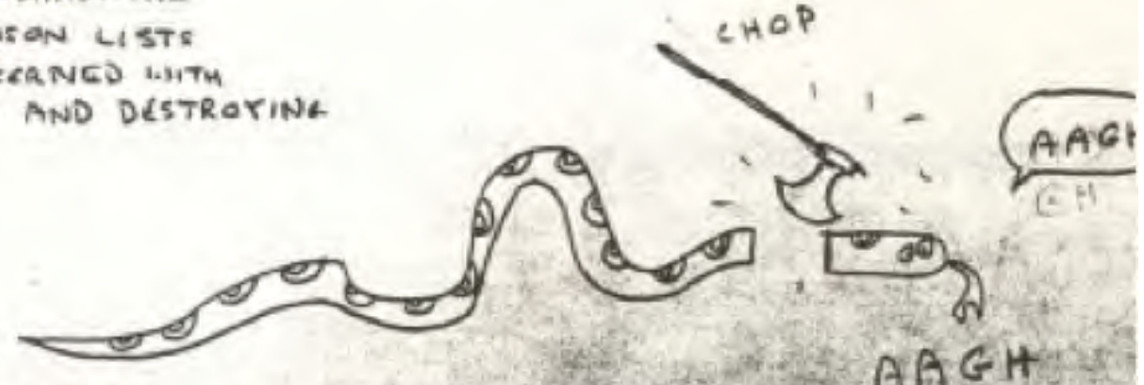
FUN WITH LISTS

PART I

A **LIST** IS A STRUCTURE THAT IS USED FOR REPRESETING A VARIABLE NUMBER OF TERMS. LISTS ARE WRITTEN BY ENCLOSING THE ITEMS (WHICH ARE SEPARATED BY COMMAS) BETWEEN [AND] brackets

X = [1, 3, Apple, pie, likes(john, sweets), 79]

THE FUNDAMENTAL OPERATIONS ON LISTS ARE CONCERNED WITH BUILDING AND DESTROYING LISTS



LISTS HAVE HEADS AND TAILS (LIKE SNAKES!)

A LIST WITH HEAD

Documentation

erlang vsn 1.05

h	help
reset	reset all queues
reset_erlang	kill all erlang definitions
load(F)	load erlang file <F>.erlang
load	load the same file as before
load(?)	what is the current load file
what_erlang	list all loaded erlang files
go	reduce the main queue to zero
send(A,B,C)	perform a send to the main queue
send(A,B)	perform a send to the main queue
cq	see queue - print main queue
wait_queue(N)	print wait_queue(N)
cf	see frozen - print all frozen states
eqns	see all equations
eqn(N)	see equation(N)
start(Mod,Goal)	starts Goal in Mod
top	top loop run system
q	quit top loop
open_dots(Node)	opens Node
talk(N)	N=1 verbose, =0 silent
peep(M)	set peeping point on M
no_peep(M)	unset peeping point on M
vsn(X)	erlang vsn number is X

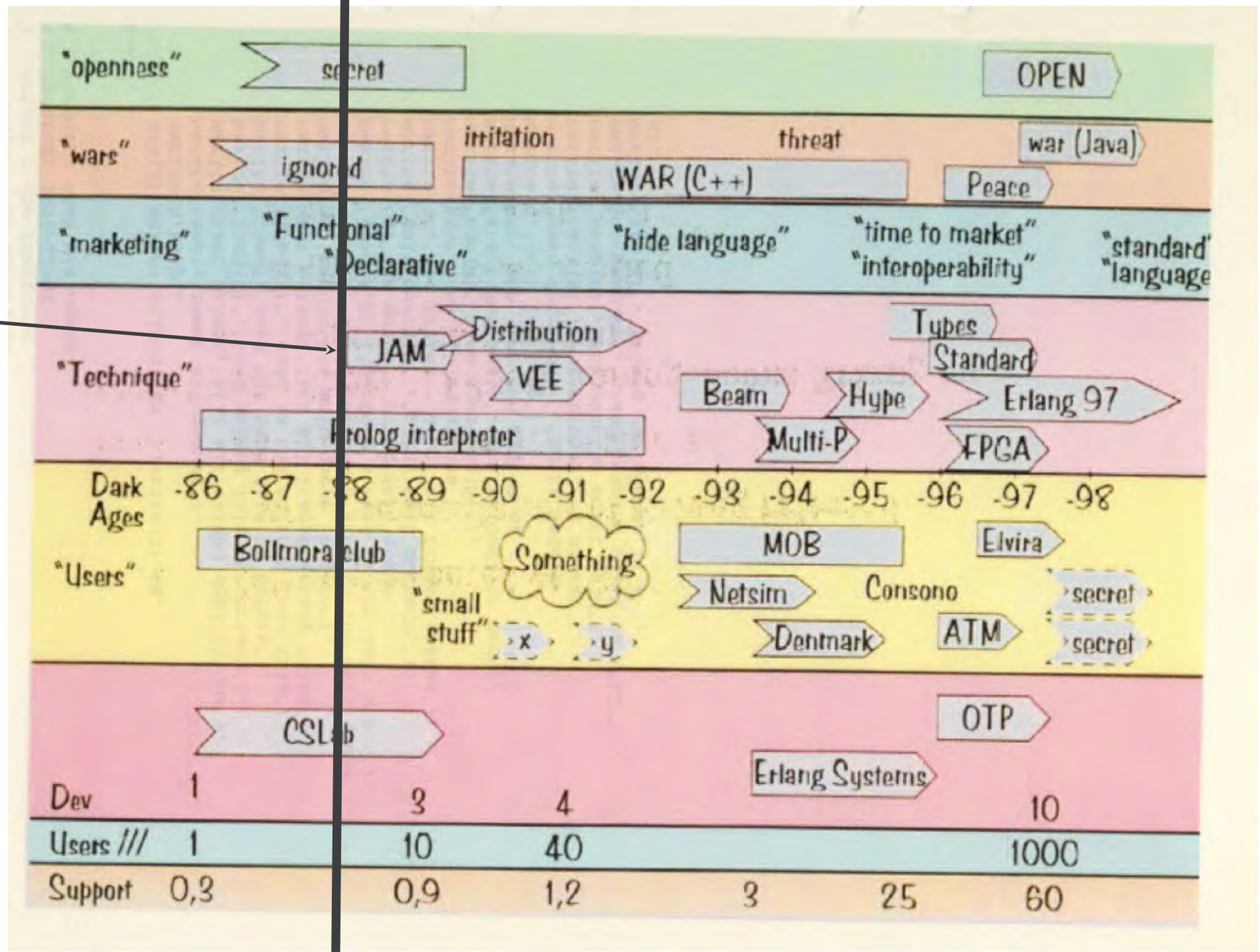
```
%   Package: User dialogues
%   Author : Joe Armstrong
%   Updated: 1986-12-04
%   Purpose: user dialogues


h:-
        vsn(X),write('erlang vsn '),write(X),nl,
        mwrite(h,help),
        mwrite(reset,'reset all queues'),
        mwrite(reset_erlang,'kill all erlang definitions'),
        mwrite('load(F)','load erlang file <F>.erlang'),
        mwrite('load','load the same file as before'),
        mwrite('load(?)','what is the current load file'),
        mwrite(what_erlang,'list all loaded erlang files'),
        mwrite(go,'reduce the main queue to zero'),
        mwrite('send(A,B,C)','perform a send to the main queue'),
        mwrite('send(A,B)','perform a send to the main queue'),
        mwrite(cq,'see queue - print main queue'),
        mwrite('wait_queue(N)','print wait_queue(N)'),
        mwrite(cf,'see frozen - print all frozen states'),
        mwrite(eqns,'see all equations'),
        mwrite('eqn(N)','see equation(N)'),
        mwrite('start(Mod,Goal)','starts Goal in Mod'),
        mwrite(top,'top loop run system'),
        mwrite(q,'quit top loop'),
        mwrite('open_dots(Node)','opens Node'),
        mwrite('talk(N)','N=1 verbose, =0 silent'),
        mwrite('peep(M)','set peeping point on M'),
        mwrite('no_peep(M)','unset peeping point on M'),
        mwrite('vsn(X)','erlang vsn number is X'),
        nl.
```

Documentation

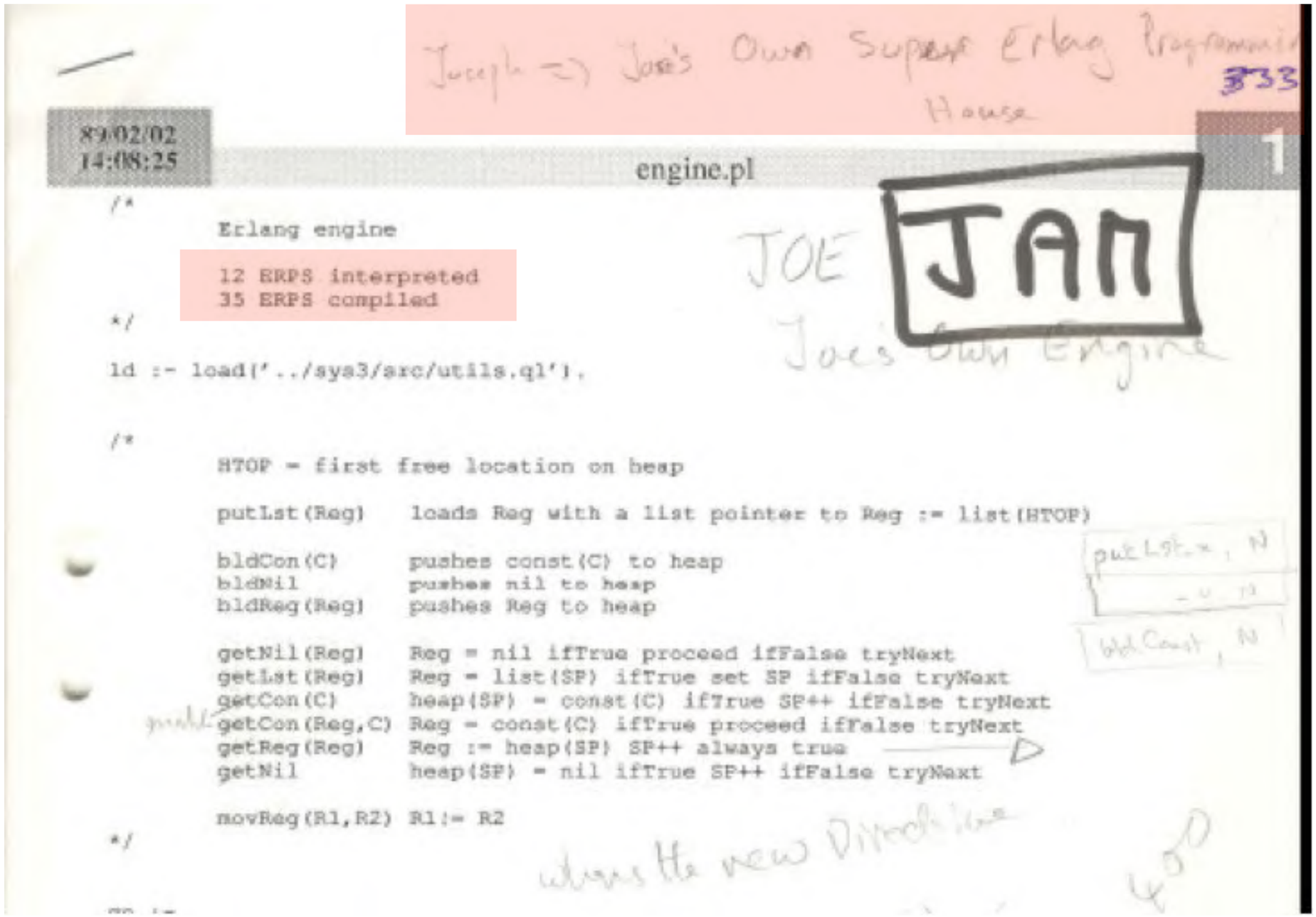
1) “Read the code.”
2) If the code and documentation differ the code is correct.
3) Flip the order of the above.
4) If the code and documentation differ the code is buggy please file an error report.
5) Things you find in the code that are not documented do not exist.
6) A few pages of markdown != Documentation.
7) Get a technical author.

PERFORMANCE


JAM


"openness"
secret
OPEN
"wars"
ignored
irritation
threat
WAR (C++)
war (Java)
Peace
"marketing"
"Functional"
"Declarative"
"hide language"
"time to market"
"interoperability"
"standard"
"language
"Technique"
JAM
Distribution
VEE
Prolog interpreter
Beam
Hype
Multi-P
Types
Standard
Erlang 97
FPGA
Dark Ages
-86 -87 -88 -89 -90 -91 -92 -93 -94 -95 -96 -97 -98
"Users"
Bollmora club
Something
"small stuff"
x
y
MOB
Netsim
Denmark
Consono
ATM
Elvira
secret
secret
CSLab
Erlang Systems
OTP
Dev 1 3 4 10
Users /// 1 10 40 1000
Support 0,3 0,9 1,2 3 25 60

1989 - speed



89/02/02
14:08:25

engine.pl

1

```
/*
        Erlang engine

        12 ERPS interpreted
        35 ERPS compiled
*/

ld :- load('../sys3/src/utils.ql').

/*
        HTOP = first free location on heap

        putLst(Reg)     loads Reg with a list pointer to Reg := list(HTOP)

        bldCon(C)       pushes const(C) to heap
        bldNil          pushes nil to heap
        bldReg(Reg)     pushes Reg to heap

        getNil(Reg)     Reg = nil ifTrue proceed ifFalse tryNext
        getLst(Reg)     Reg = list(SP) ifTrue set SP ifFalse tryNext
        getCon(C)       heap(SP) = const(C) ifTrue SP++ ifFalse tryNext
        getCon(Reg,C)   Reg = const(C) ifTrue proceed ifFalse tryNext
        getReg(Reg)     Reg := heap(SP) SP++ always true
        getNil          heap(SP) = nil ifTrue SP++ ifFalse tryNext

        movReg(R1,R2)   R1:= R2
*/
```

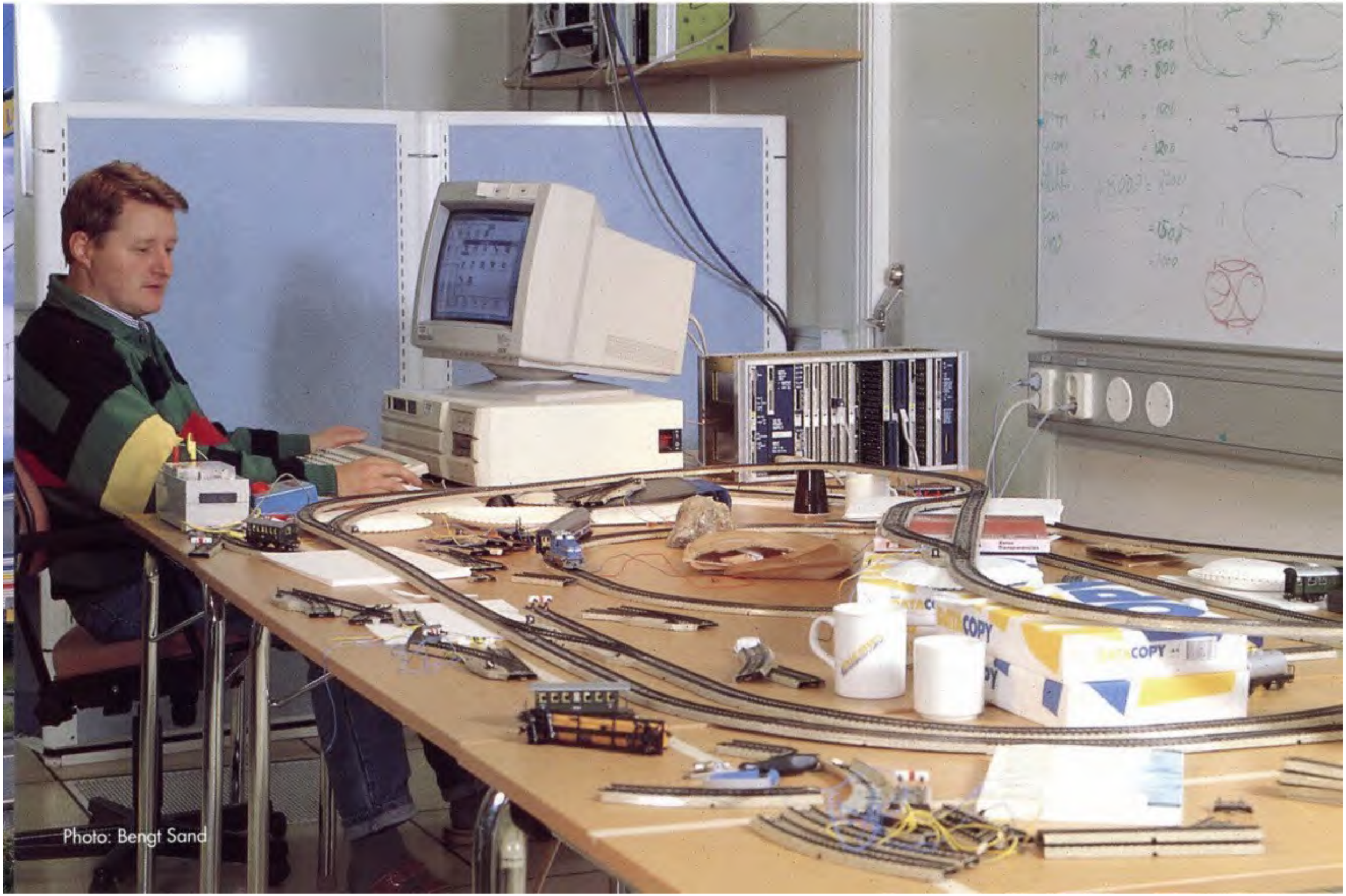


Photo: Bengt Sand

Performance

- Wait 10 years for x1000 improvement.
- Incorrect code can be made arbitrarily fast.
- Erlang is millions of times faster than in 1986 – virtually all the speedups come from hardware not better software (MHz clock → GHz clock)
- Projects are cancelled because the software does not work.
- Can always make stuff faster by throwing hardware it it.

Joe's predictions

- In the future (say 20 years from now) all performance critical software will be synthesized into hardware. It's the energy.

1992 - 1995

nothing much happens, ...

8 Dec 1995

AXE-N Cancelled ...

1996

AXD 301 starts

Lot's of stuff happens quickly

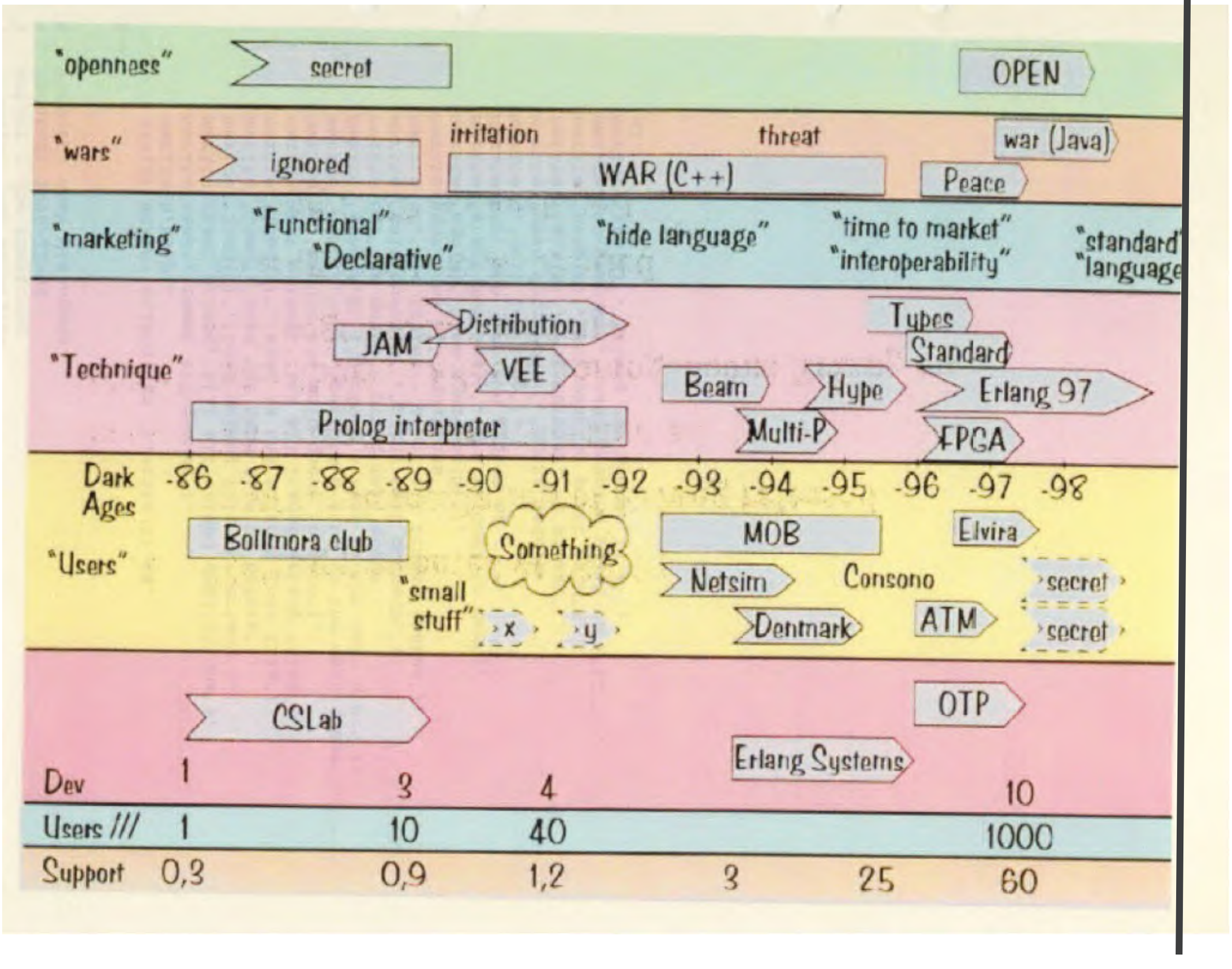
1996 - 1998

nothing much happens ...

1998

AXD 301 is a great success so we are ...

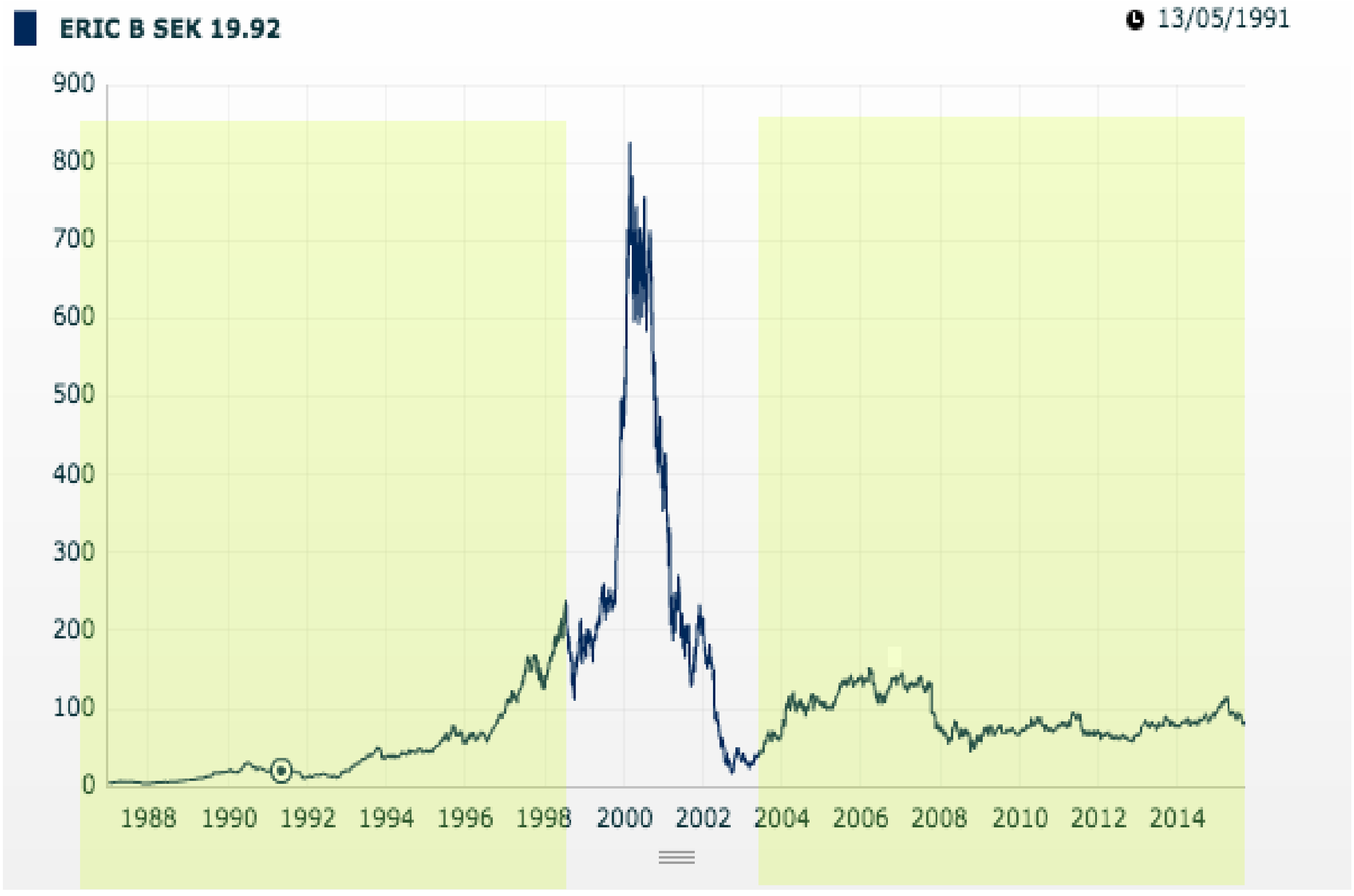
BANNED


"openness"
secret
OPEN
"wars"
ignored
irritation
WAR (C++)
threat
war (Java)
Peace
"marketing"
"Functional"
"Declarative"
"hide language"
"time to market"
"interoperability"
"standard"
"language
"Technique"
JAM
Distribution
VEE
Prolog interpreter
Beam
Hype
Multi-P
Types
Standard
Erlang 97
FPGA
Dark Ages
-86 -87 -88 -89 -90 -91 -92 -93 -94 -95 -96 -97 -98
"Users"
Bollmora club
Something
"small stuff"
x
y
MOB
Netsim
Denmark
Consono
ATM
Elvira
secret
secret
CSLab
OTP
Erlang Systems
Dev 1 3 4 10
Users /// 1 10 40 1000
Support 0,3 0,9 1,2 3 25 60

1998 Stuff Happens

- Lots of things happen very quickly
- Erlang becomes Open Source
- 4 days later we quit and form Bluetail

- But there is a back story … something is happening ...


ERIC B SEK 19.92
13/05/1991
900
800
700
600
500
400
300
200
100
0
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014

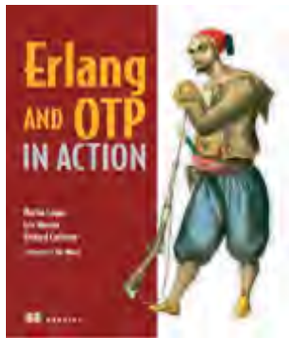
10 years later...


Concurrent Programming in ERLANG
ERLANG


Erlang
Programming
O'REILLY


Programming Erlang
Joe Armstrong


Erlang
语言编程
O'REILLY


Erlang AND OTP IN ACTION


プログラミング Erlang


GOTOP
Erlang程式設計

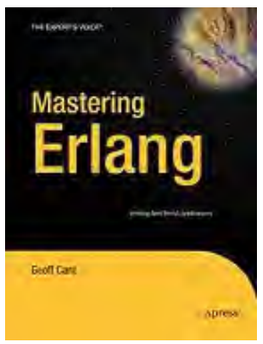

Concurrent Programming in ERLANG
Joe Armstrong
Robert Virding
Mike Williams


Building Web Applications with Erlang
Early Release
O'REILLY


Erlang
programmation
EYROLLES


Erlang/OTP


Erlang
プログラミング
O'REILLY


Mastering
Erlang
Geoff Cant
Apress


プログラミング言語
Erlang 入門
ASCII

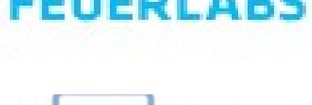

Learn You Some Erlang For Great Good!


Programming Erlang


Programmer en Erlang

Who is Using Erlang?






- Teaching materials
- Community
- Courses
- Conferences
- Users
- Companies
- Technology