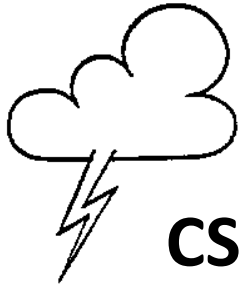
CSLab and all that …

Bjarne Däcker Tekn lic, Tekn dr h c

- Ericsson employee 1966 – 2002
- Manager CSLab 1984 – 2002
- Now retired



CSLab

Computer Science Laboratory at Ericsson

- 1980 informally started
- 1984 formally established
- 1997 – 2001 **SARC** (Software Architecture Research Centre) spun off from CSLab
- 2002 closed down

Thus 10 – 15 people during 20 years → 200 – 300 manyears !!!

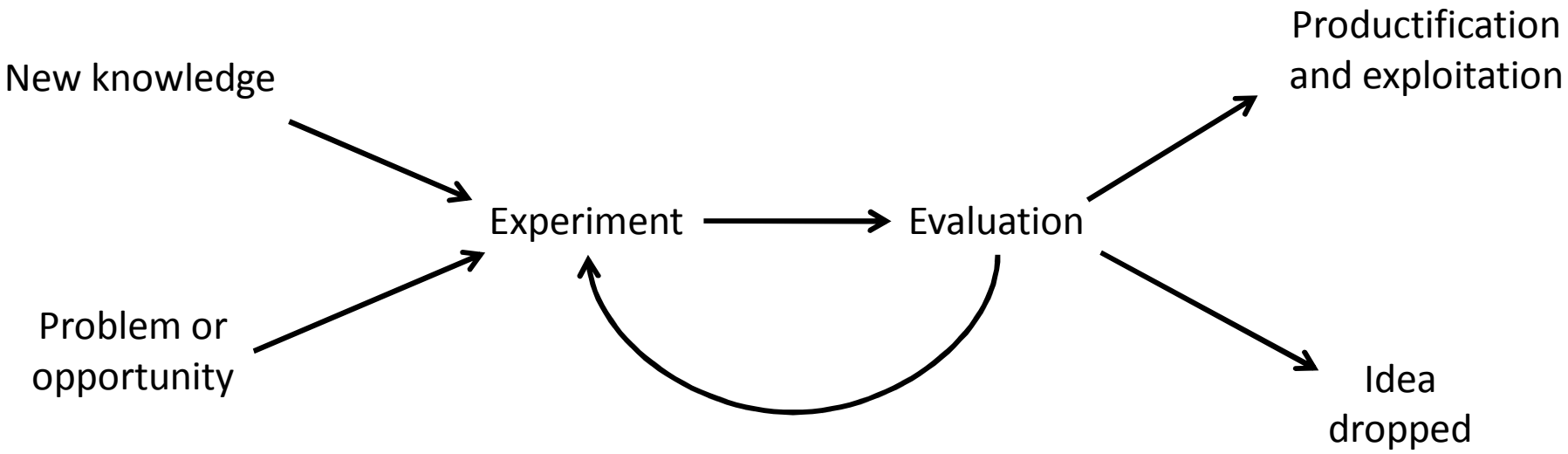
Basic Research

- Discover or create new knowledge for mankind

Applied Research

- Apply new knowledge to problems or applications in the real world

The Process of Applied Research



Can be problem driven or technology driven

Terrible lesson learnt only too late:
Marketing is all important. People might not believe (in) something they see working but might well be prepared to trust something that they have only heard of.

If it works for other technologies, why not for this software as well…
but might well be prepared…

CSLab charter

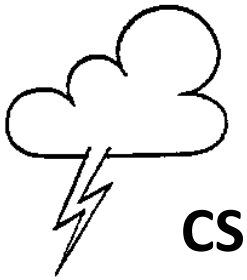
- Develop software technology for future telecom systems and support systems

- In the near term contribute to the introduction of new technology in existing systems

April 19, 1984

Mike Williams' credo

- Find the right methods – design by prototyping
- Make mistakes in a small scale – not in a production project
- It is not good enough to have ideas – you must also be able to implement them yourself to know that they work



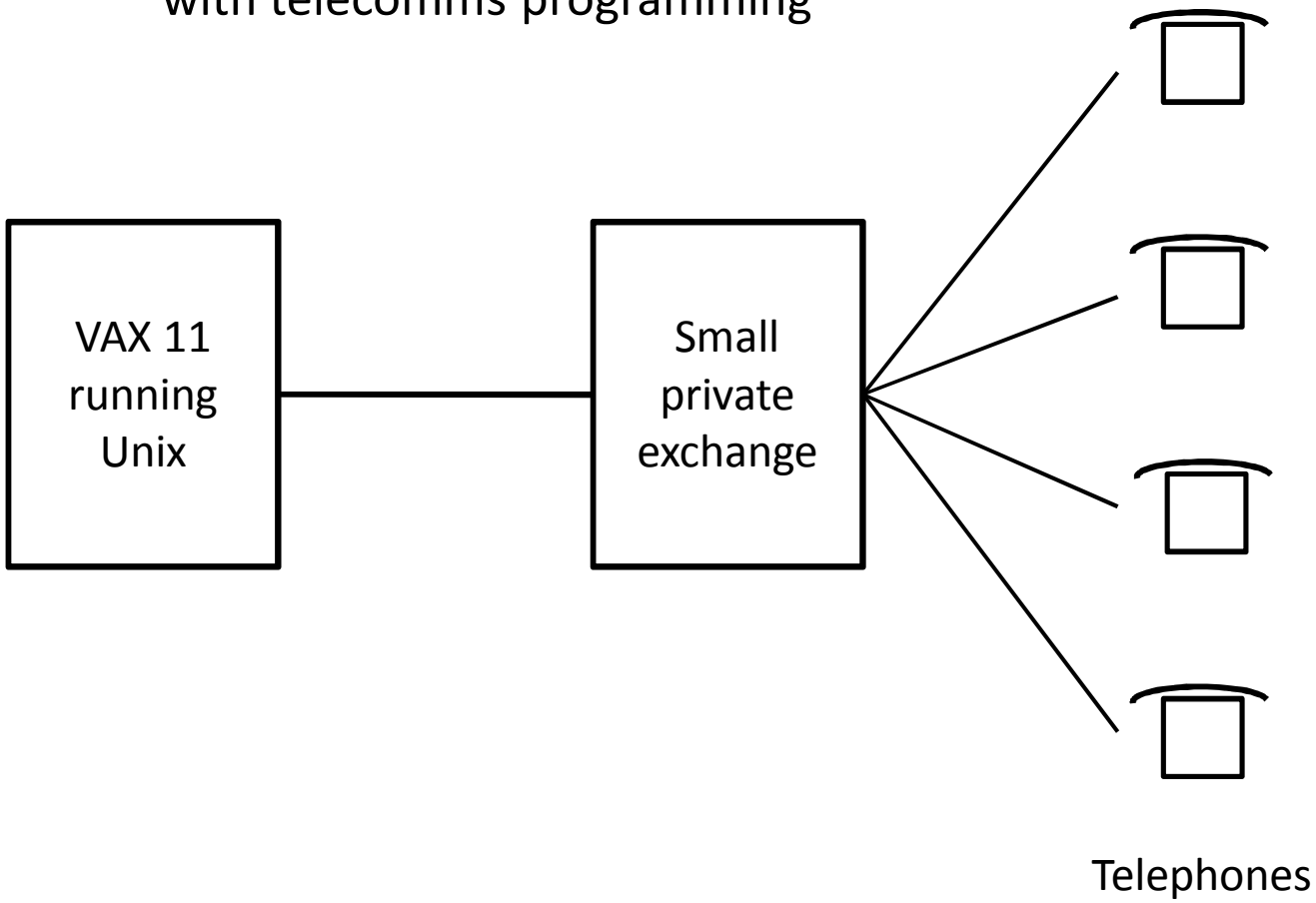
CSLab’s lucky timing

- The Japanese Fifth Generation project
- The European Esprit project
- The British Alvey project

Some other projects at CSLab

- Unix
- TCP/IP
- Transputer
- RISC Architectures
- Super conductors (!)
- Smalltalk computer
- Prolog, Lisp
- Work stations
- Graphics user interface
- Expert Systems
- IP access over passive cable networks

Laboratory environment for experiments
with telecomms programming



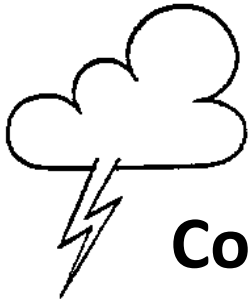
Techniques tried

- Imperative Programming Languages
  - Concurrent Euclid
  - Ada
- Declarative Programming Languages
  - PFL (Parallel Functional Language)
  - LPL (Logic Programming Language)
- Rules Based Systems
  - OPS4
- Object Oriented Languages
  - Frames
  - CLU

Conclusions 1

- Telecoms can apparently be programmed in any language
- A small language like Concurrent Euclid managed very well
- The process concept is very useful
- Concepts like buffers or rendez-vous are very awkward
- Functional languages are powerful but need some database
- Logic languages and rule based systems give a nice declarative approach but need modularisation
- Object oriented languages handle modularity but need concurrency

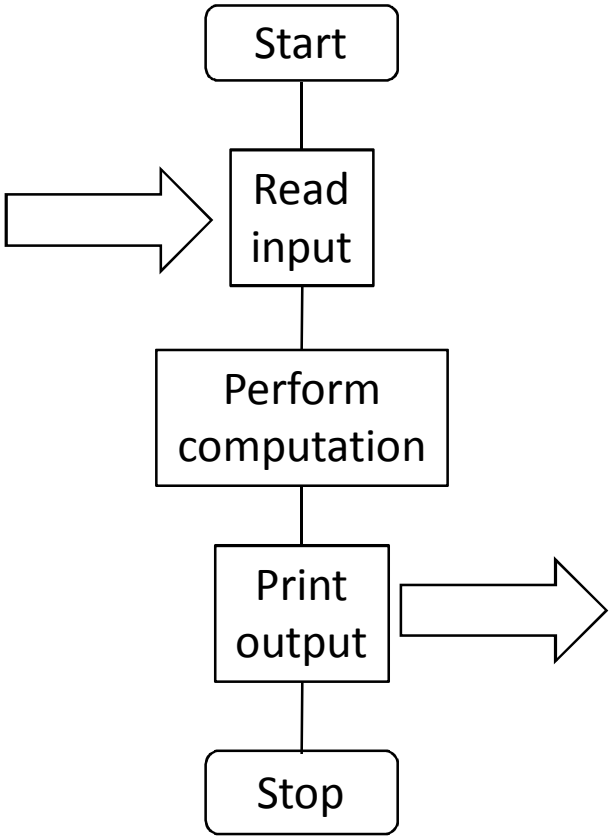
January 1984



Conclusions 2

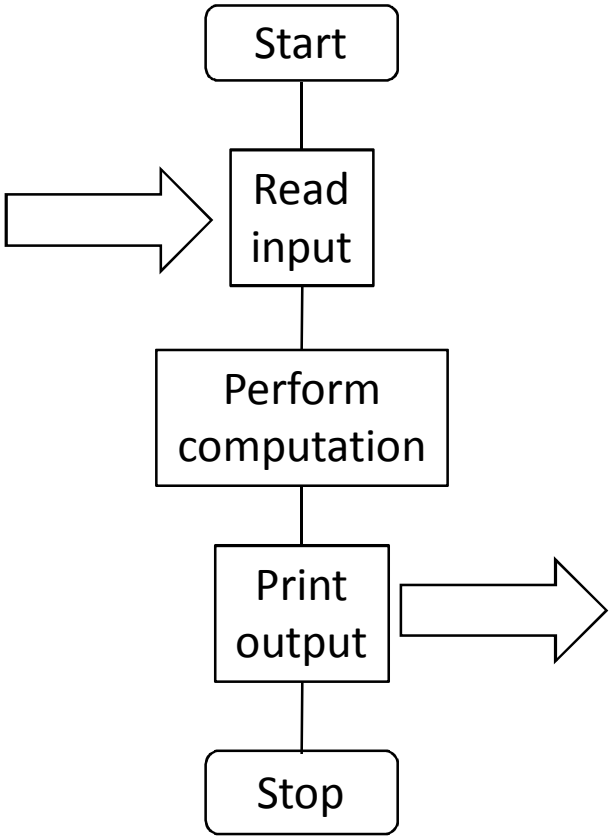
- Dealing with concurrency in imperative languages by means of process abstraction and the methods for process interwork are now established technologies
- Adding concurrency to CLU could probably be done in a fairly conventional manner…
- Future systems will probably be built up using many of the techniques used in these experiments, for example expert systems for maintenance functions, logic programming for programming signal system interfaces and the underlying OS might be programmed in an advanced imperative language

A typical Pascal program (?)



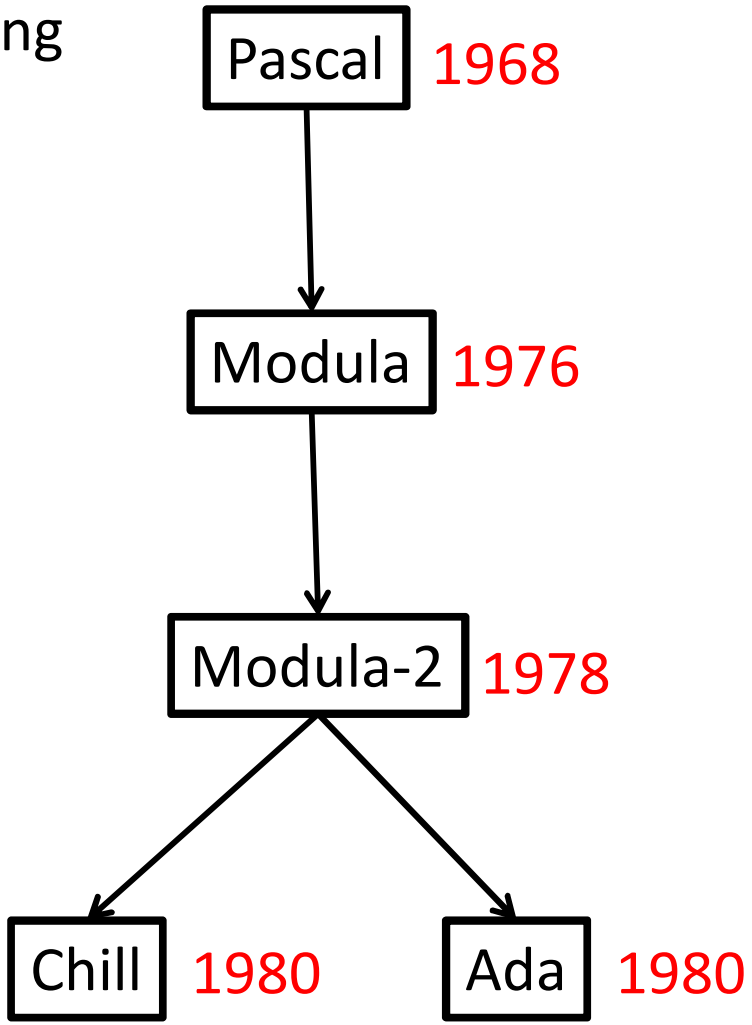
What more is required when used for designing large real-time control systems ?

A typical Pascal program (?)



- A **module** concept for structuring a large program system being designed by many people
- A **process** concept to describe concurrent activities
- A process communication concept
- Means to communicate with hardware etc. etc.

Some systems programming languages



Modula and Modula-2

Designed by Nikolaus Wirth

A module concept **module**

A process concept **process**

Process communication by using shared variables in **interface modules**

Ada

Designed by a committee under Jean Ichbiah
ordered by the US Dept of Defense

A module concept **package**

A process concept **task**

Process communication by *rendez-vous*, i.e. one task calls a procedure in another task with synchronization

Chill

Designed by a committee working under C.C.I.T.T.

A module concept **module**

A process concept **process**

Three methods for process communication

- **regions** like Modula's interface modules
- **buffers** like mailboxes where processes can deposit and retreive messages
- **signals** which are messages sent directly from one process to another

When there are several messages waiting it is
undefined which one that will be received

EriPascal

Designed at Ericsson intended to be equivalent to a subset of Chill

A module concept **module**

A process concept **process** corresponding to a normal Pascal program

Process communication by **signals** and *selective receive* like in Erlang (except no pattern matching)

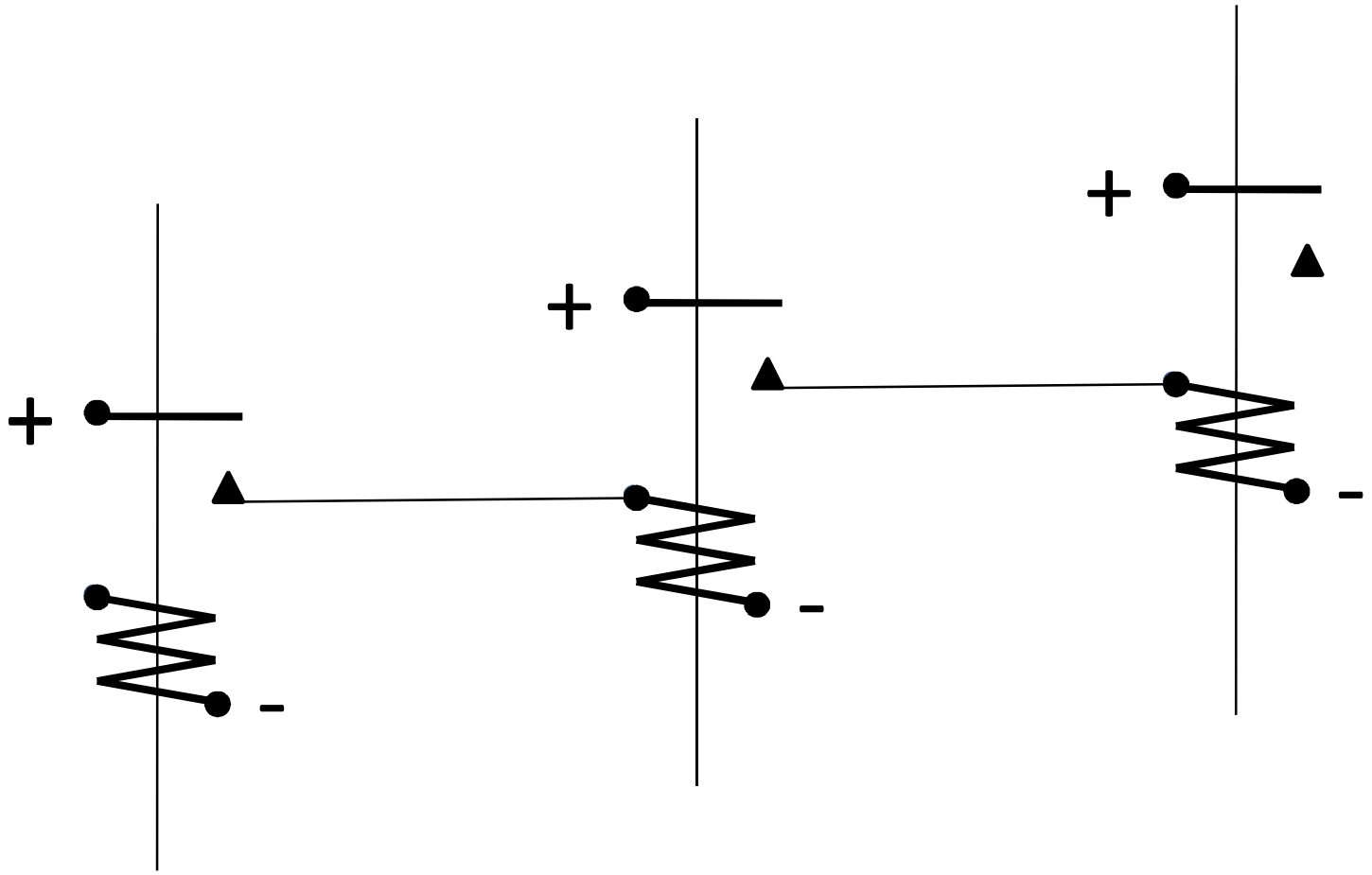
Some further observations

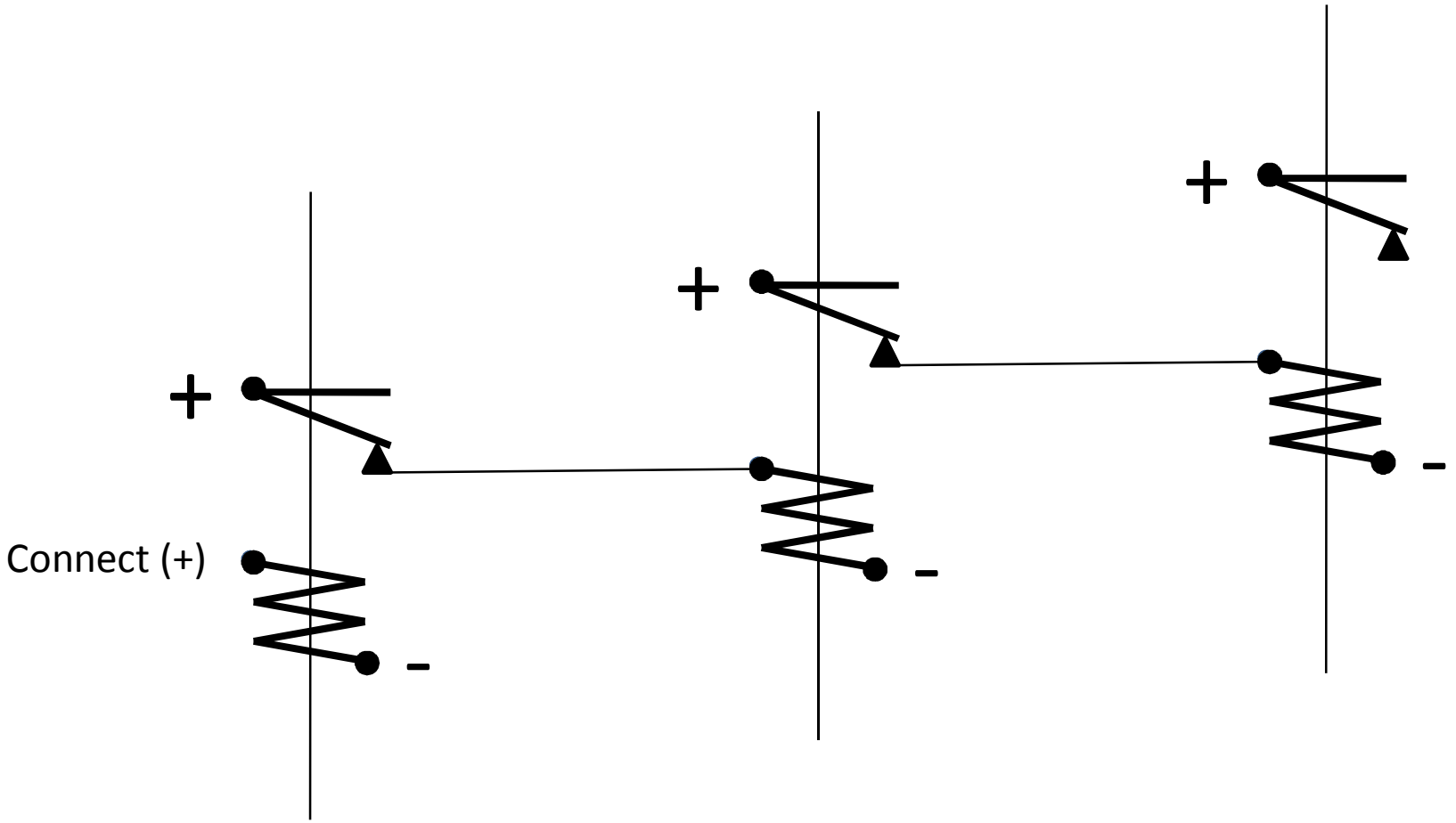
The process concept is very useful to model concurrency but is too heavyweight in most languages to implement massive concurrency

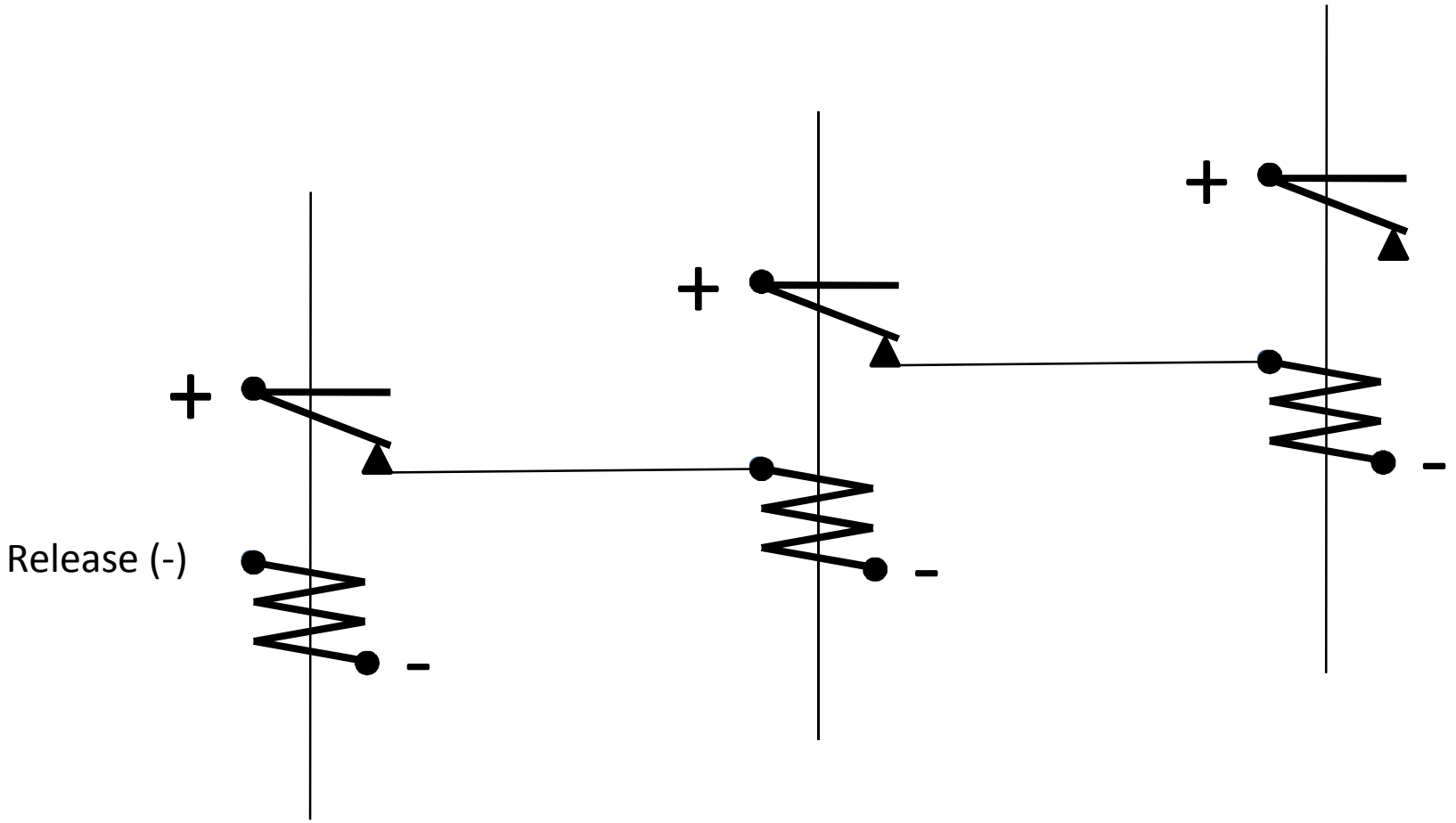
Fault handling on a per process basis and the ability to connect (link) processes so that one process can supervise another. Default being that if one process crashes all linked processes also crash but with the possibility to override this default and "trap" the exit message. This idea comes from the way old relay equipment used to work (C-wire).

Ability to replace code in a running system. This not only reduces down time, but also facilitates debugging and correction of errors

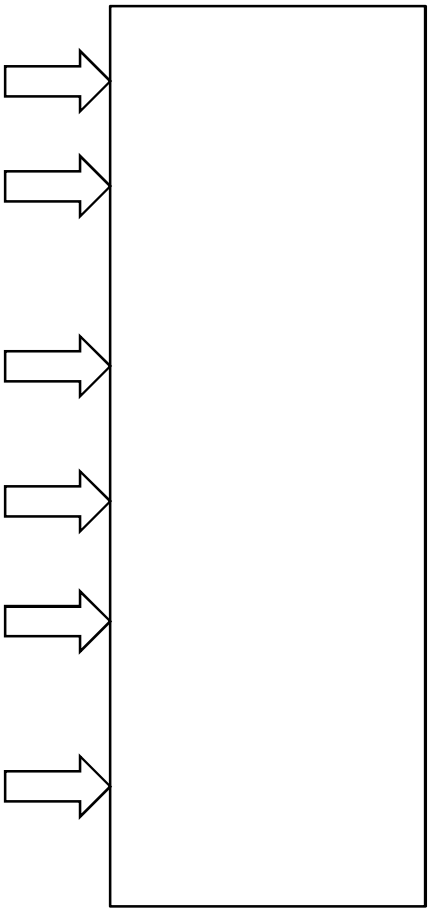
Some good old-fashioned relay set technology




+
+
+
Connect (+)
-
-
-


+
+
+
Release (-)
-
-
-

Signal-state or state-signal ?



Cooperating automatons are popular in telecomms

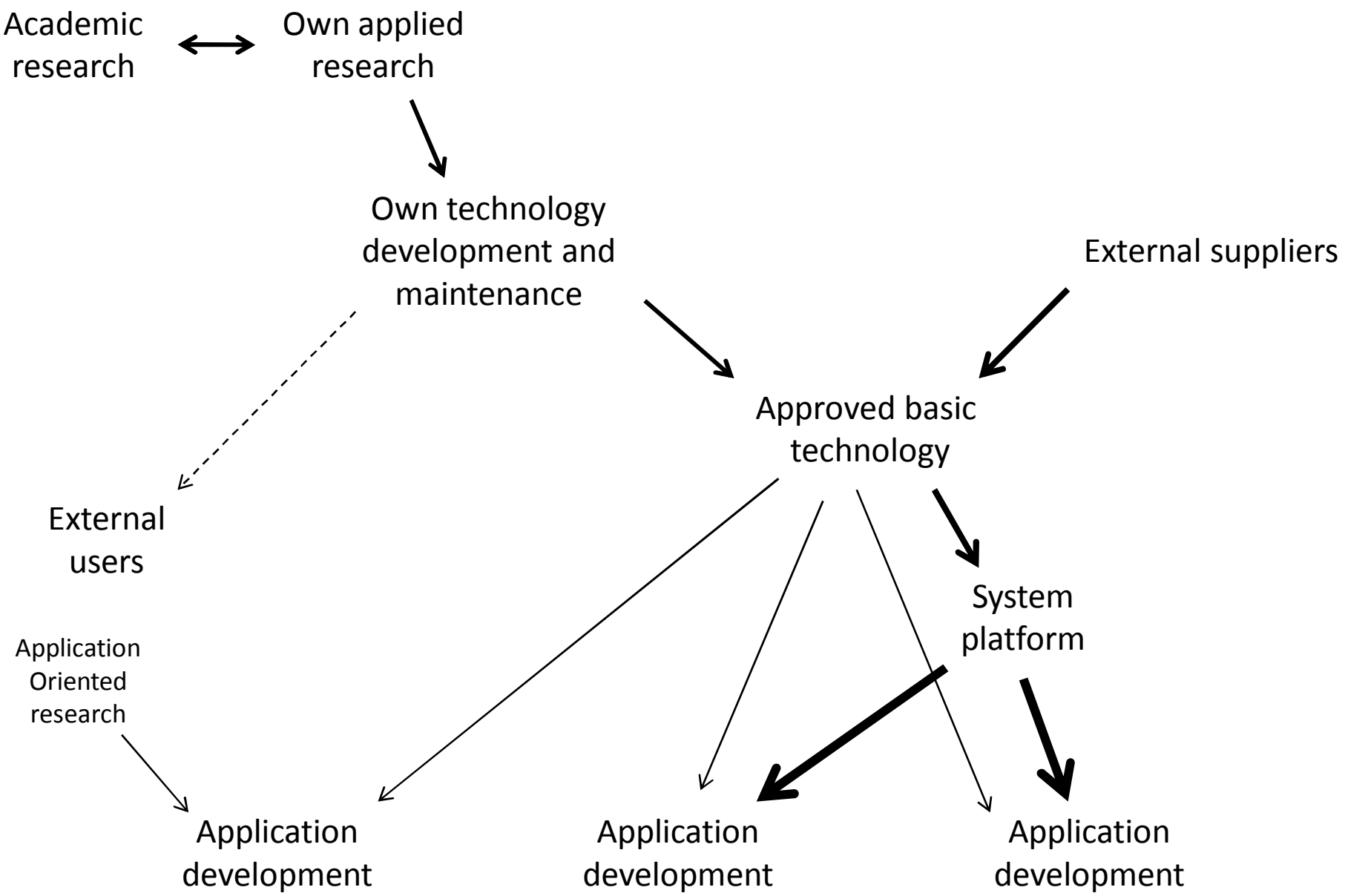
The figure shows the programming model of a PLEX program for AXE with entry points for different signals

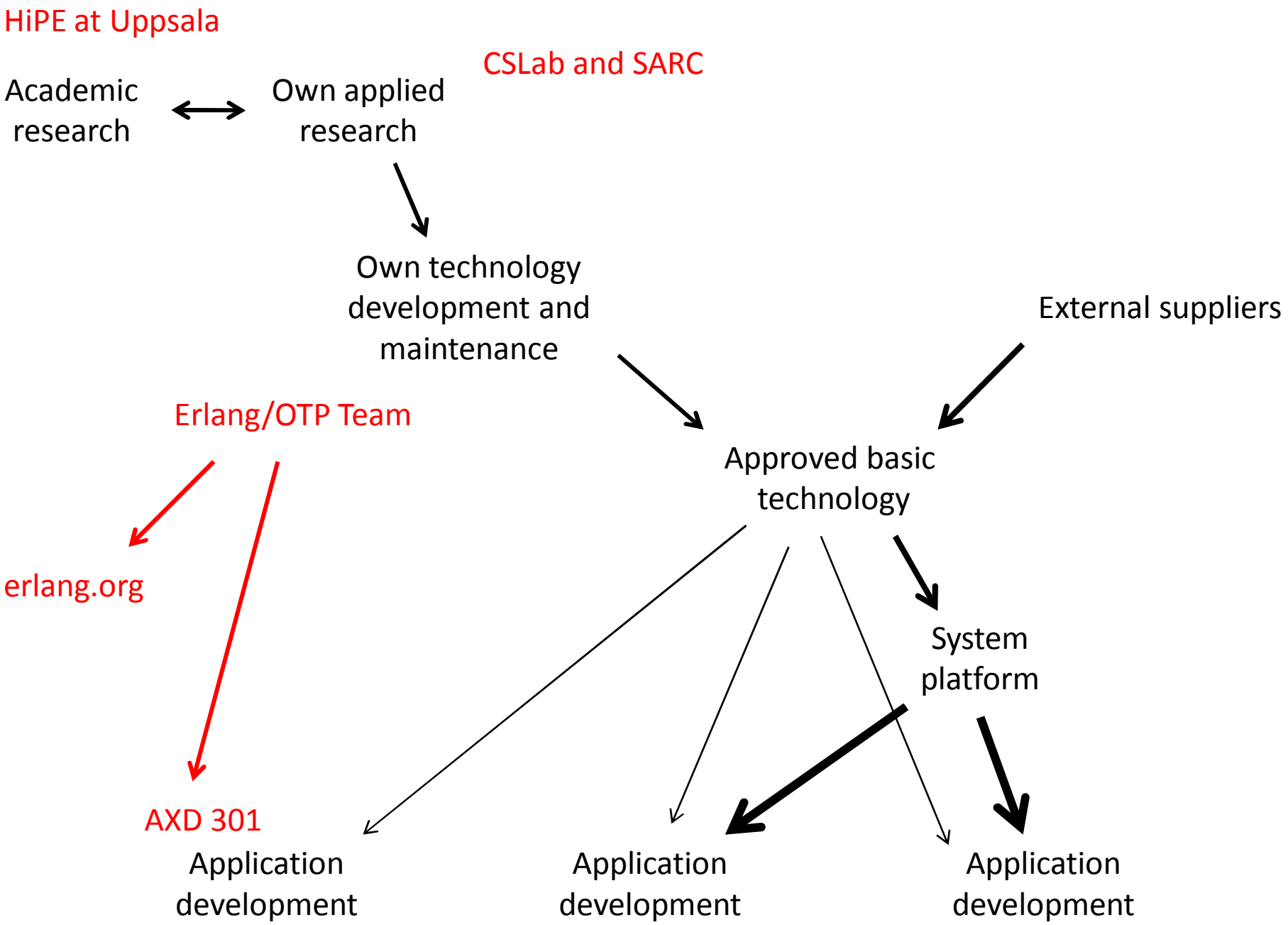
The programmer has to administer the program flow himself

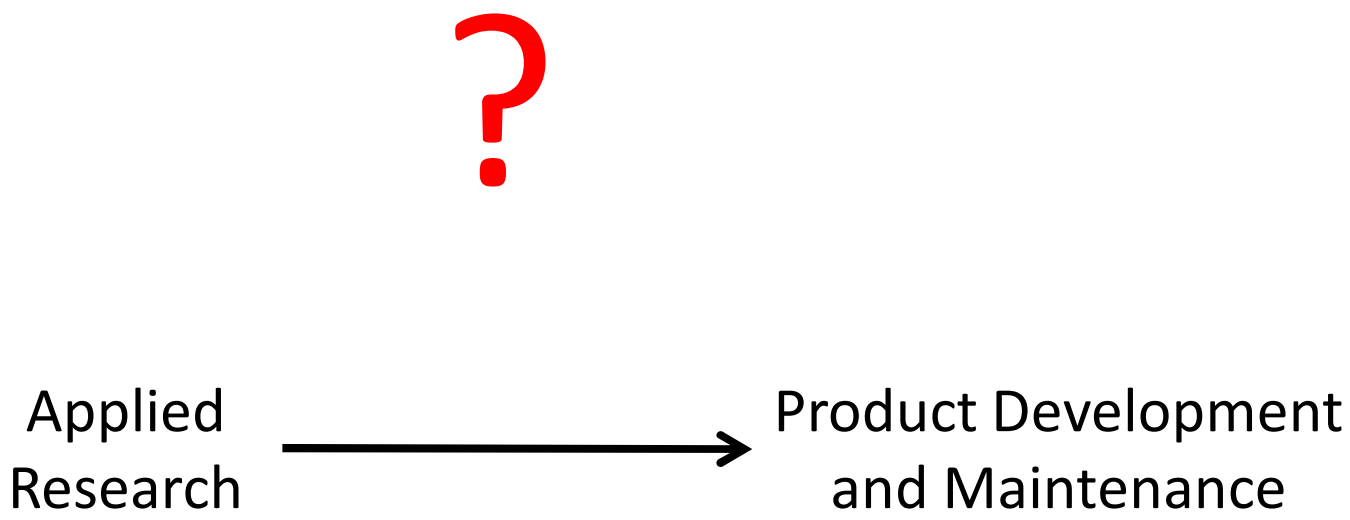
Signals arriving when not expected require special action

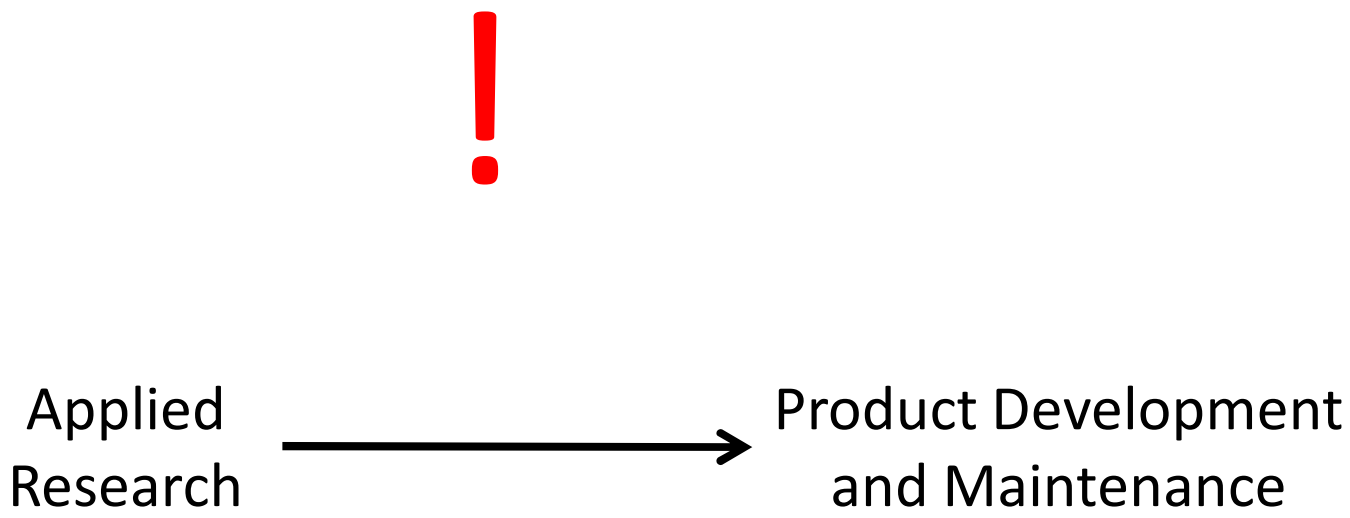
Language manual

- EriPascal – An internal Ericsson report 1984
- Erlang – A book printed by Prentice Hall 1993


Academic research
Own applied research
Own technology development and maintenance
External suppliers
Approved basic technology
External users
System platform
Application Oriented research
Application development
Application development
Application development


HiPE at Uppsala
CSLab and SARC
Academic research
Own applied research
Own technology development and maintenance
External suppliers
Erlang/OTP Team
Approved basic technology
erlang.org
System platform
AXD 301
Application development
Application development
Application development

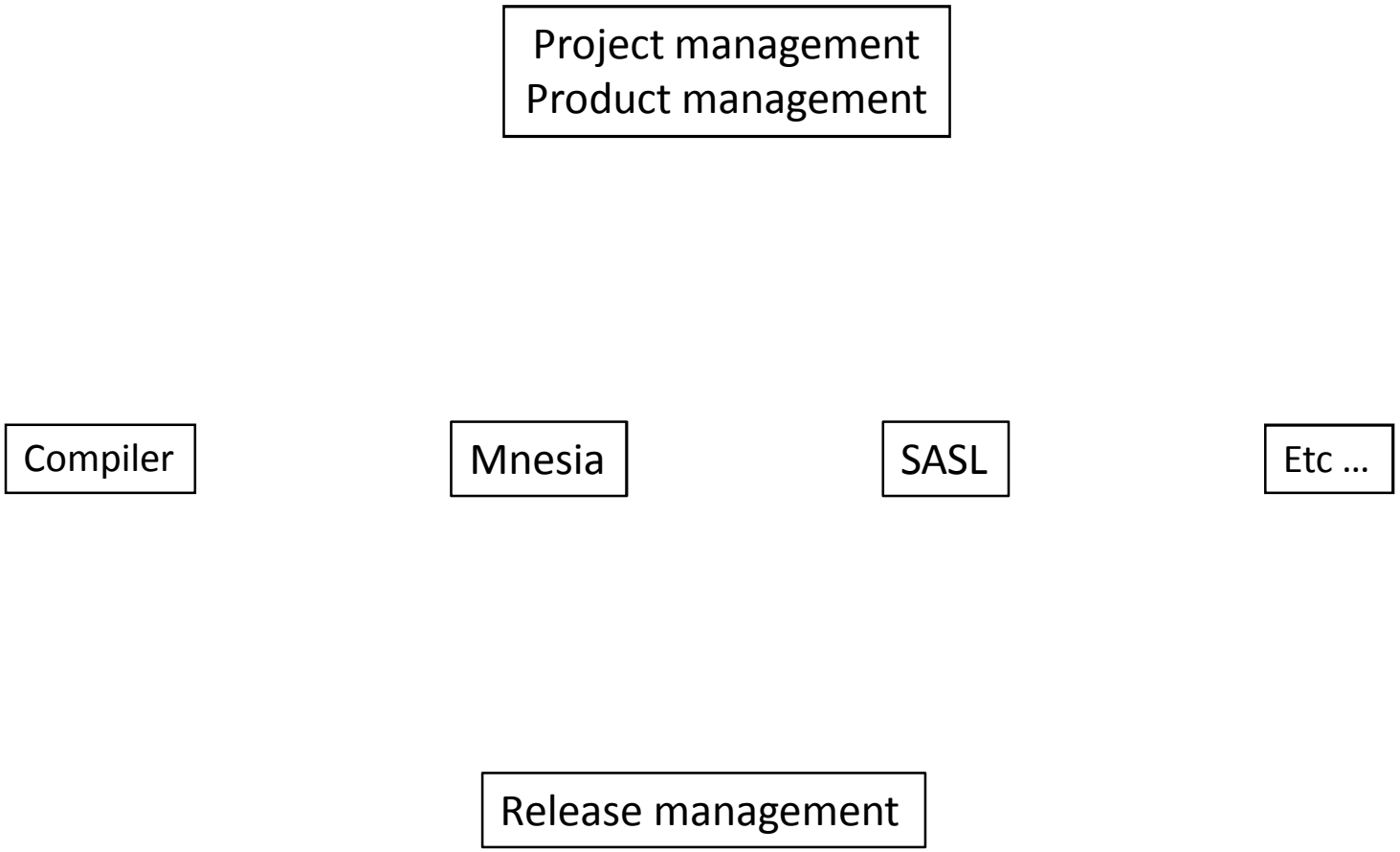

?
Applied Research
Product Development and Maintenance



Some proposed strategies

- Throw it over the wall and see what happens
- Move the people

Technology transfer



Technology transfer

Phase 1

Project management
Product management

CSLab

Compiler

CSLab + OTP

Mnesia

CSLab + OTP

SASL

CSLab + OTP

Etc …

CSLab + OTP

Release management

OTP

Technology transfer

Phase 2

Project management
Product management

OTP

Compiler

CSLab + OTP

Mnesia

CSLab + OTP

SASL

CSLab + OTP

Etc …

CSLab + OTP

Release management

OTP

Technology transfer

Phase 3 …

Project management
Product management

OTP

Compiler

OTP + CSLab

Mnesia

OTP + CSLab

SASL

OTP + CSLab

Etc …

OTP + CSLab

CSLab successively phased out

Release management

OTP

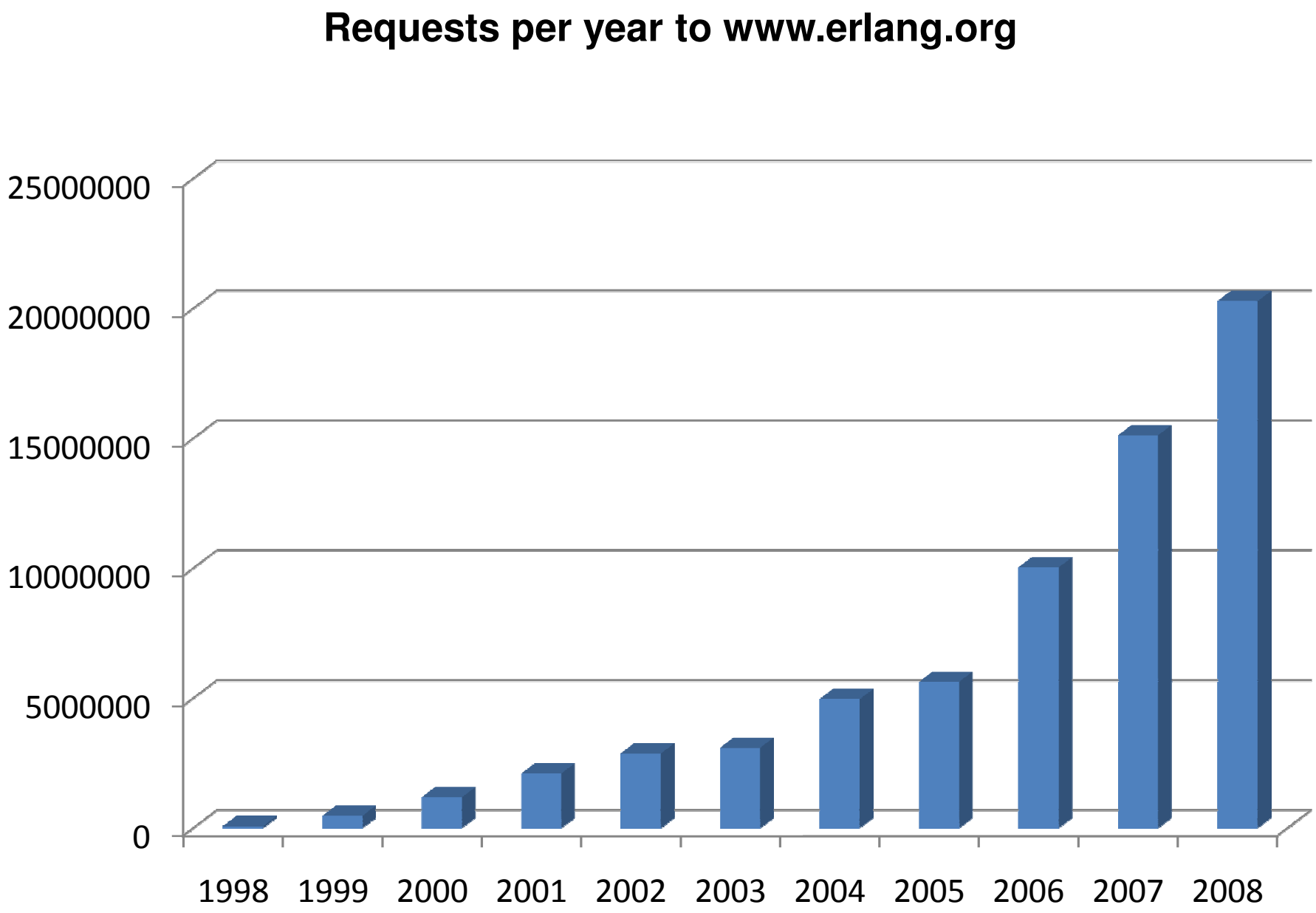
Early Erlang history

	Internal usage	External usage
1984-86	Technology evaluations	-
1987-89	Use in prototypes	-
1990-92		Academic distribution
1993-95	Limited use in products	External marketing
1996	Use for strategic product development	External marketing stopped
1997	OTP team created	External marketing restarted
1998	Nine products displayed at CeBit	3,323 evaluation systems delivered
1998	Erlang banned at ERA for new products	Open source release
1999….	AXD301 and GPRS win important orders	Growing use for product development

Management intervenes …

The importance of chance and individuals

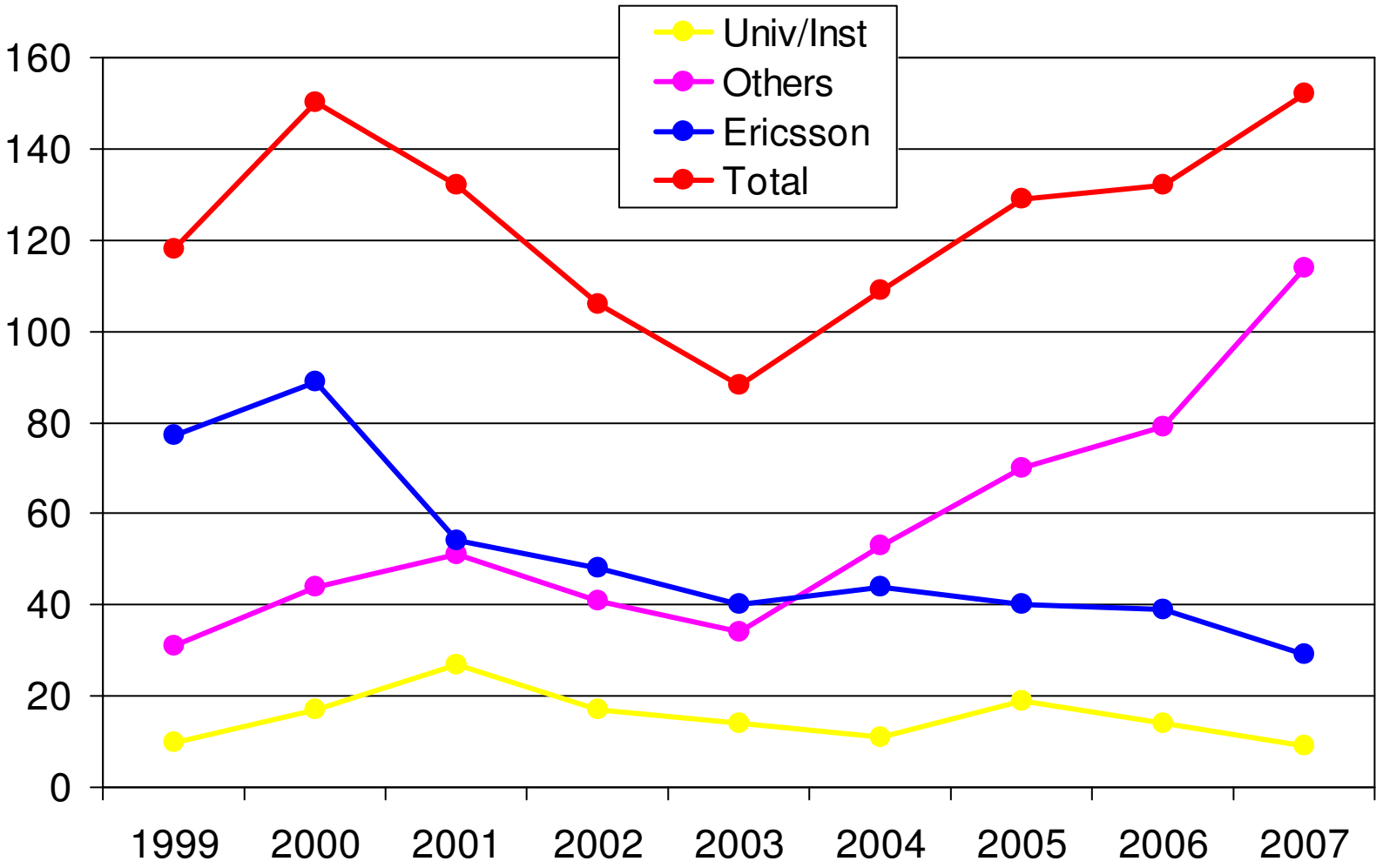
- Mike Williams coming from within Ericsson
- Joe Armstrong coming from outside Ericsson
- Bogdan Hausman coming from SICS
- Claes Wikström creating distributed Erlang and Mnesia
- Thomas Lindgren initiating HiPE
- Jane Walerud getting approval for Open Source
- Francesco Cesarini setting up Erlang Training & Consulting
- Kenneth Lundin ensuring a professional quality product
- Mickaël Rémond writing a French book on Erlang
- etc. etc. etc. etc.


Requests per year to www.erlang.org
25000000
20000000
15000000
10000000
5000000
0
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008

Erlang/OTP International User Conferences
are held every year in Älvsjö

This photo shows the audience of the 11th EUC in 2005 with more than 130 participants

EUC participation

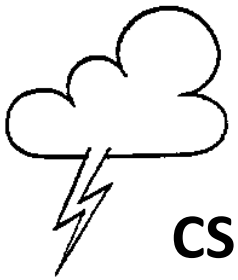


ACM SIGPLAN Erlang Workshops

have been held since 2002 in connection with ICFP International Conference on Functional Programming

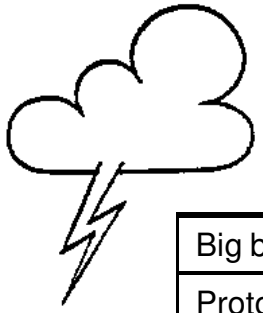
The ACM SIGPLAN workshops were held 2002 in Pittsburgh, 2003 in Uppsala, 2004 in Snowbird, Utah, 2005 in Tallinn, 2006 in Portland, 2007 in Freiburg and 2008 in Victoria, B.C. The photograph shows the audience of the workshop in Tallinn

The next workshop will be on September 5, 2009, in Edinburgh



CSLab after Erlang

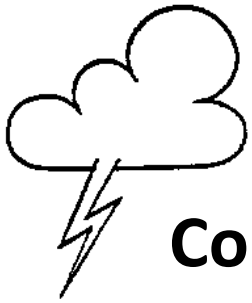
- SIP, Megaco and other protocol stuff
- Program verification
- Speech technology
- Collaboration with various Ericsson projects like home communication …



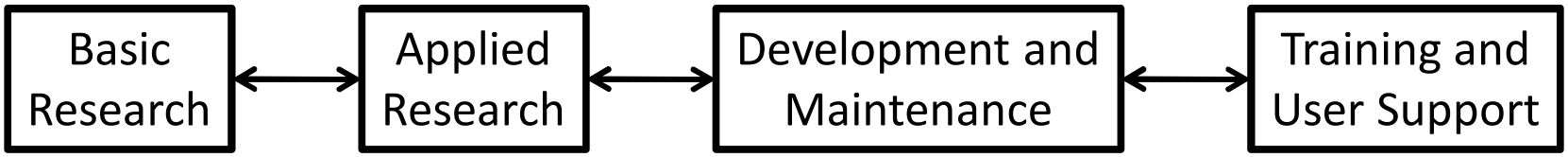
What happened to them all ?

Big boss at Ericsson	Mike Williams
Prototyping at Ericsson	Joe Armstrong
Swedish Defense Institute	Robert Virding
VINNOVA	Bogdan Hausman
	Nabiel Elshiewy
Erlang/OTP Team at Ericsson	Lars Thorsén
	Håkan Mattsson
Tail-f	Håkan Millroth
	Claes Wikström better known as Klacke
	Per Hedeland
	Sebastian Strollo
	Johan Bevemyr
Kreditor	Torbjörn Törnkvist
	Magnus Fröberg
Corelatus	Matthias Läng
Professor of Computer Science, Göteborg	Thomas Arts
Professor of Computer Architecture, Uppsala	Erik Hagersten
Board member of Erlang Training & Consulting	Bjarne Däcker

More info at the nostalgic homepage www.cs-lab.org



Conclusions



- It's a rôle game. All parts are required
- Keep them separate but in close cooperation
- Mike Williams' credo #2 – If you don't experiment, your project will be an experiment