HYATT

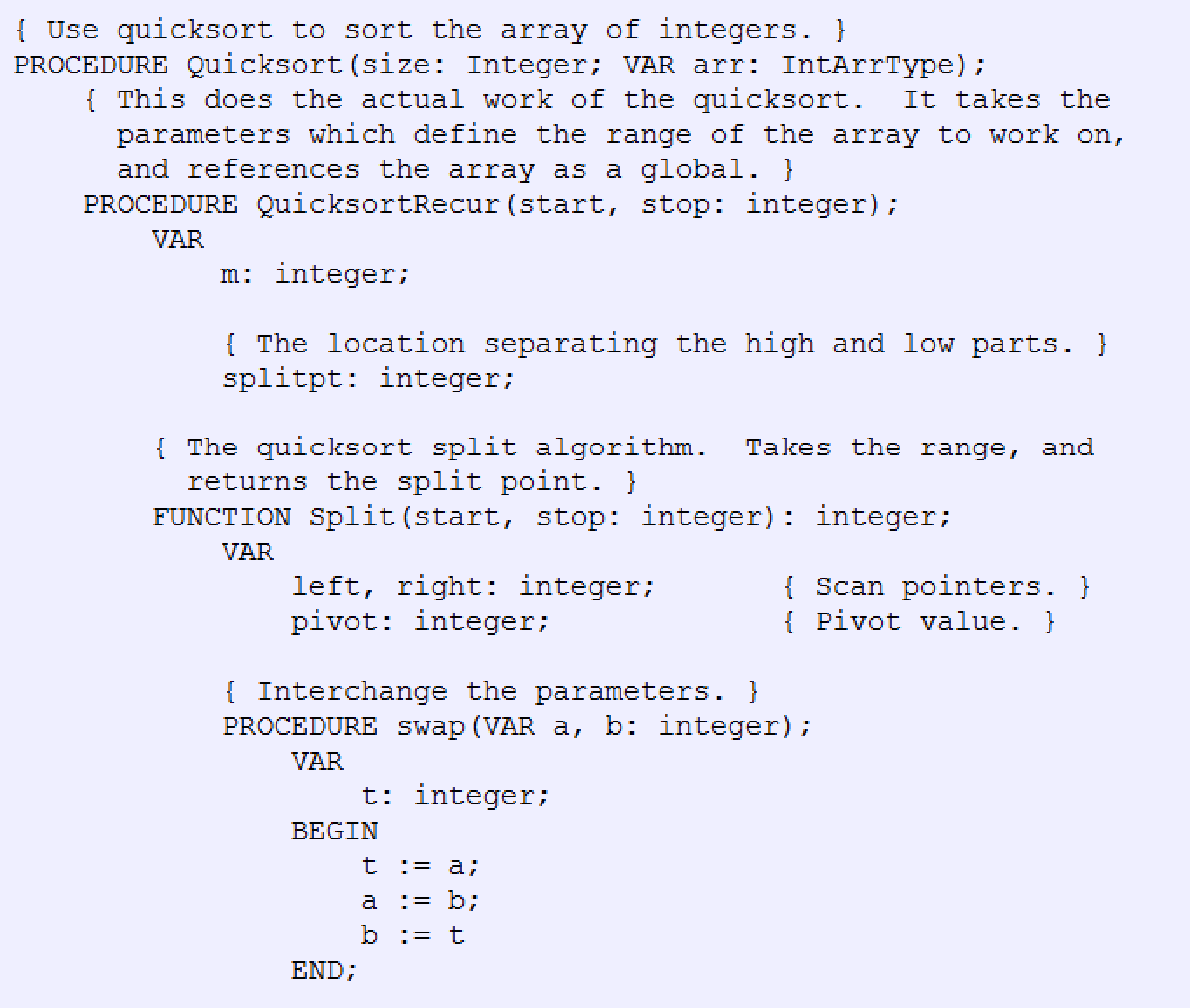
# ”Life is too short for imperative programming”

John Hughes

# Sorting in Haskell

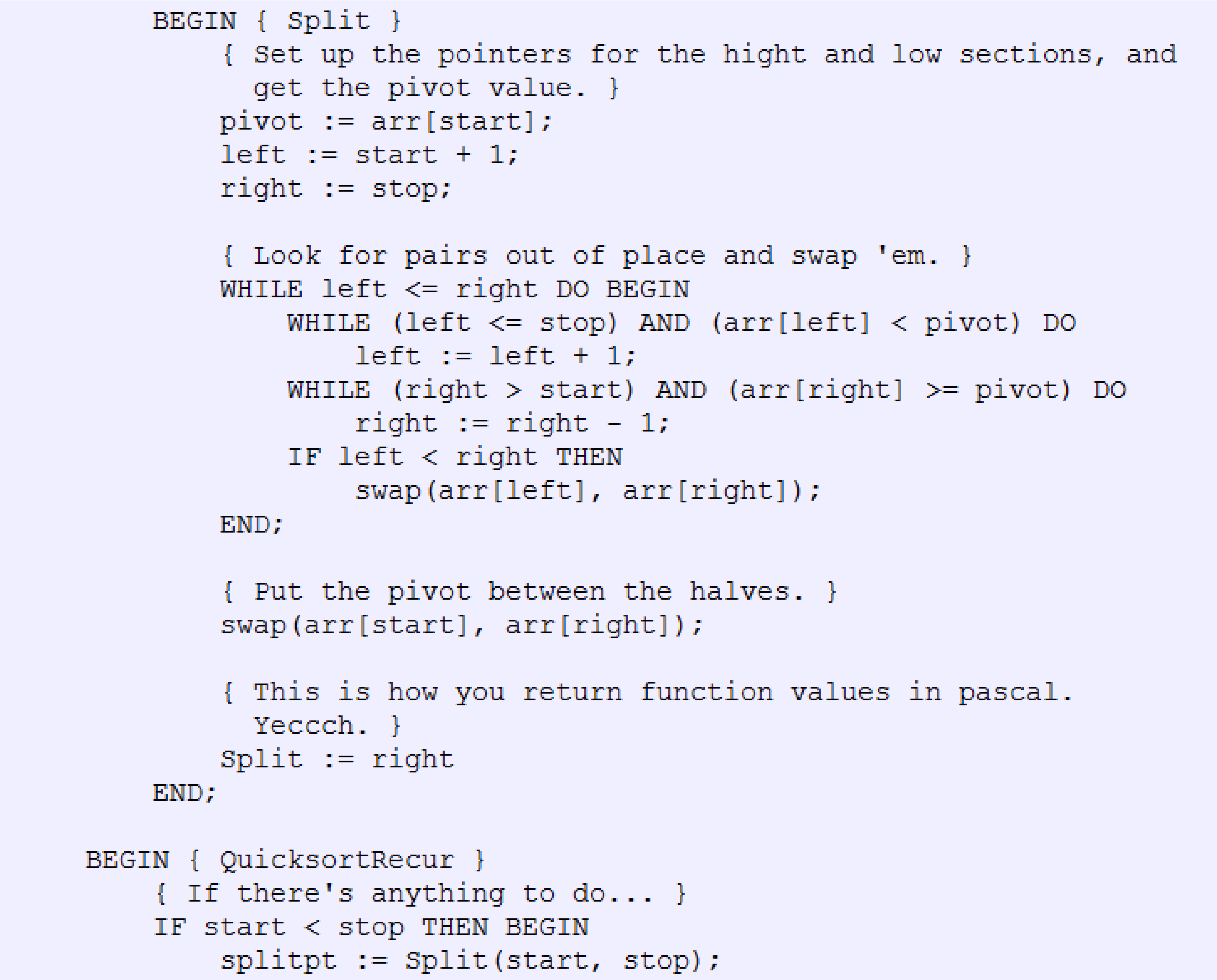
```
sort [] = []
sort (x:xs) =
     sort [y | y <- xs, y<x]
  ++ [x]
  ++ sort [y | y <- xs, y>=x]
```

# Sorting in Pascal



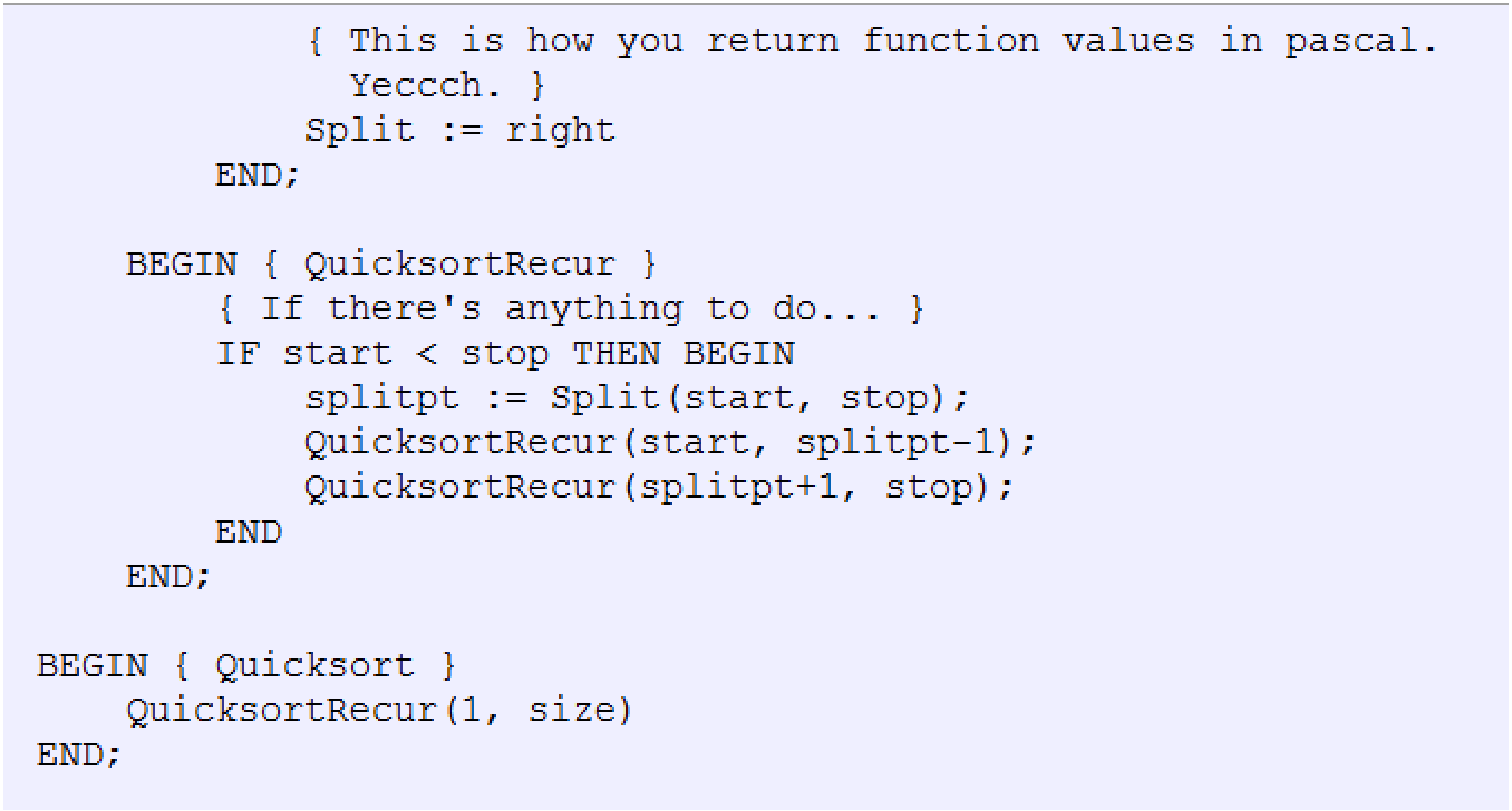
```pascal
{ Use quicksort to sort the array of integers. }
PROCEDURE Quicksort(size: Integer; VAR arr: IntArrType);
    { This does the actual work of the quicksort.  It takes the
      parameters which define the range of the array to work on,
      and references the array as a global. }
    PROCEDURE QuicksortRecur(start, stop: integer);
        VAR
            m: integer;

            { The location separating the high and low parts. }
            splitpt: integer;

        { The quicksort split algorithm.  Takes the range, and
          returns the split point. }
        FUNCTION Split(start, stop: integer): integer;
            VAR
                left, right: integer;       { Scan pointers. }
                pivot: integer;             { Pivot value. }

            { Interchange the parameters. }
            PROCEDURE swap(VAR a, b: integer);
                VAR
                    t: integer;
                BEGIN
                    t := a;
                    a := b;
                    b := t
                END;
```

# Sorting in Pascal, page 2



```
        BEGIN { Split }
            { Set up the pointers for the hight and low sections, and
              get the pivot value. }
            pivot := arr[start];
            left := start + 1;
            right := stop;

            { Look for pairs out of place and swap 'em. }
            WHILE left <= right DO BEGIN
                WHILE (left <= stop) AND (arr[left] < pivot) DO
                    left := left + 1;
                WHILE (right > start) AND (arr[right] >= pivot) DO
                    right := right - 1;
                IF left < right THEN
                    swap(arr[left], arr[right]);
            END;

            { Put the pivot between the halves. }
            swap(arr[start], arr[right]);

            { This is how you return function values in pascal.
              Yeccch. }
            Split := right
        END;

    BEGIN { QuicksortRecur }
        { If there's anything to do... }
        IF start < stop THEN BEGIN
            splitpt := Split(start, stop);
```

# Sorting in Pascal, page 3



```
            { This is how you return function values in pascal.
              Yeccch. }
            Split := right
        END;

    BEGIN { QuicksortRecur }
        { If there's anything to do... }
        IF start < stop THEN BEGIN
            splitpt := Split(start, stop);
            QuicksortRecur(start, splitpt-1);
            QuicksortRecur(splitpt+1, stop);
        END
    END;

BEGIN { Quicksort }
    QuicksortRecur(1, size)
END;
```

# Sorting in Java

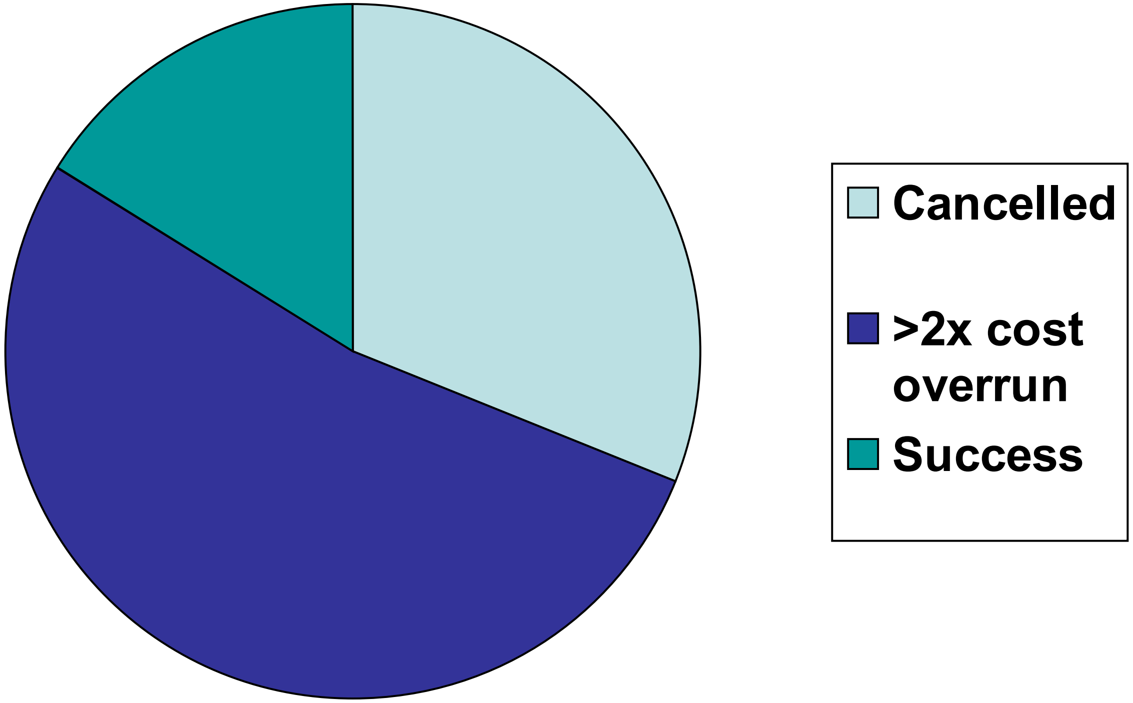
```
private void quicksort(int low, int high) {
      int i = low, j = high;
      int pivot = numbers[low + (high-low)/2];
      while (i <= j) {
            while (numbers[i] < pivot) {
                  i++;
            }
            while (numbers[j] > pivot) {
                  j--;
            }
            if (i <= j) {
                  exchange(i, j);
                  i++;
                  j--;
            }
      }
```

# Sorting in Java, page 2

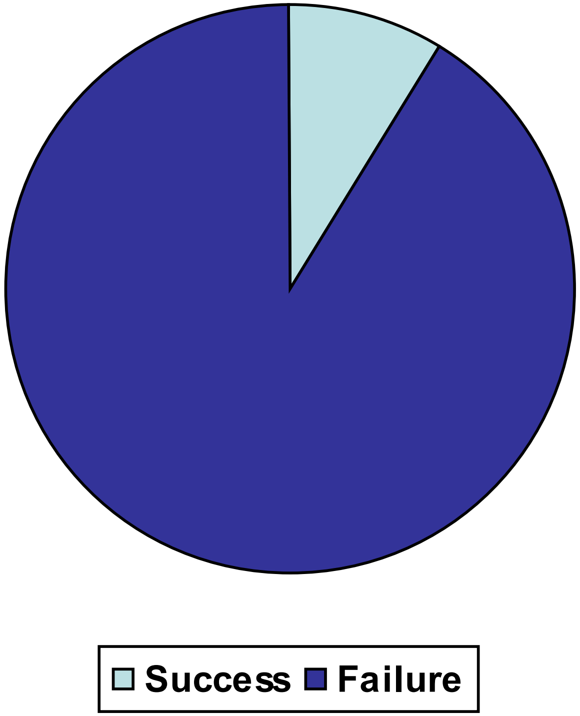
```
     if (low < j)
          quicksort(low, j);
     if (i < high)
          quicksort(i, high);
}

private void exchange(int i, int j)
{
     int temp = numbers[i];
     numbers[i] = numbers[j];
     numbers[j] = temp;
}
```

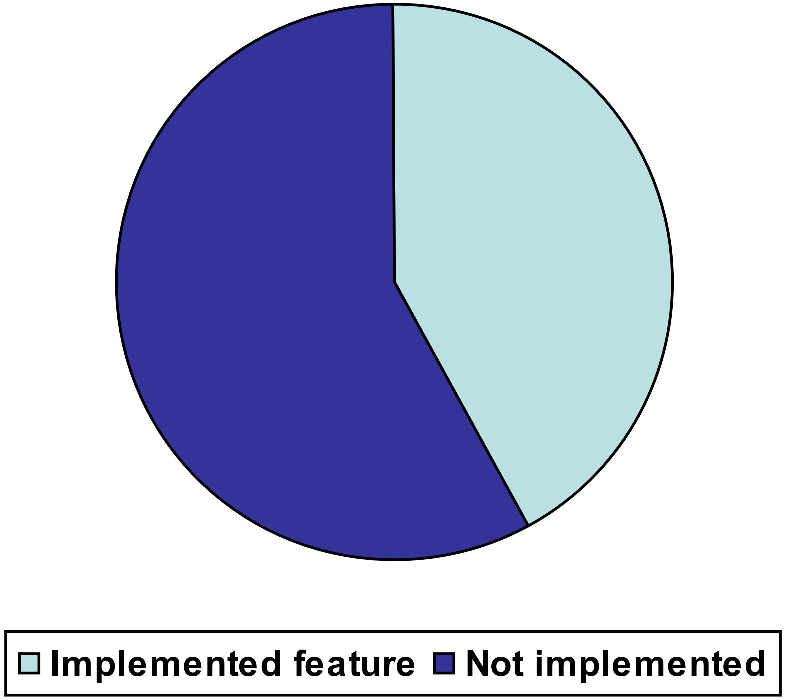
# Software Crisis, 1968—today

- Software project outcomes

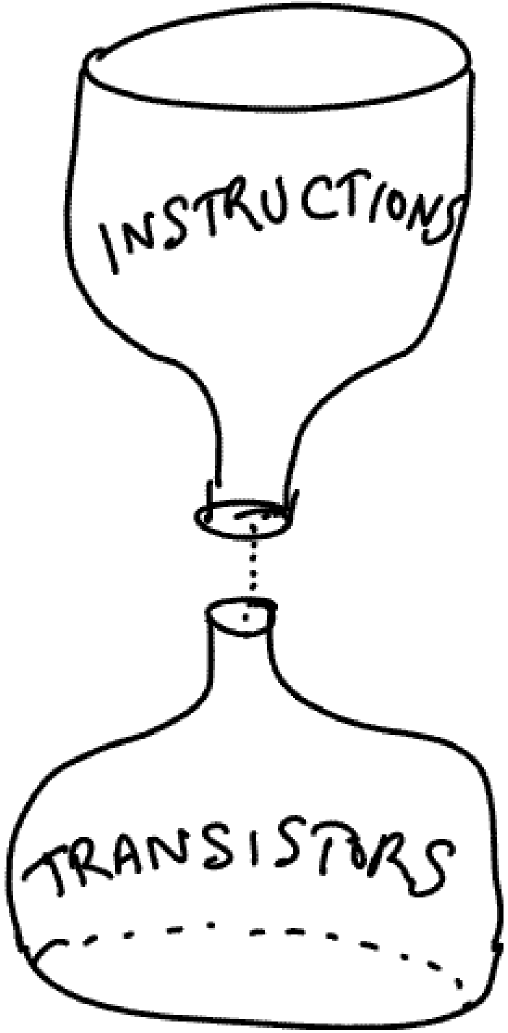


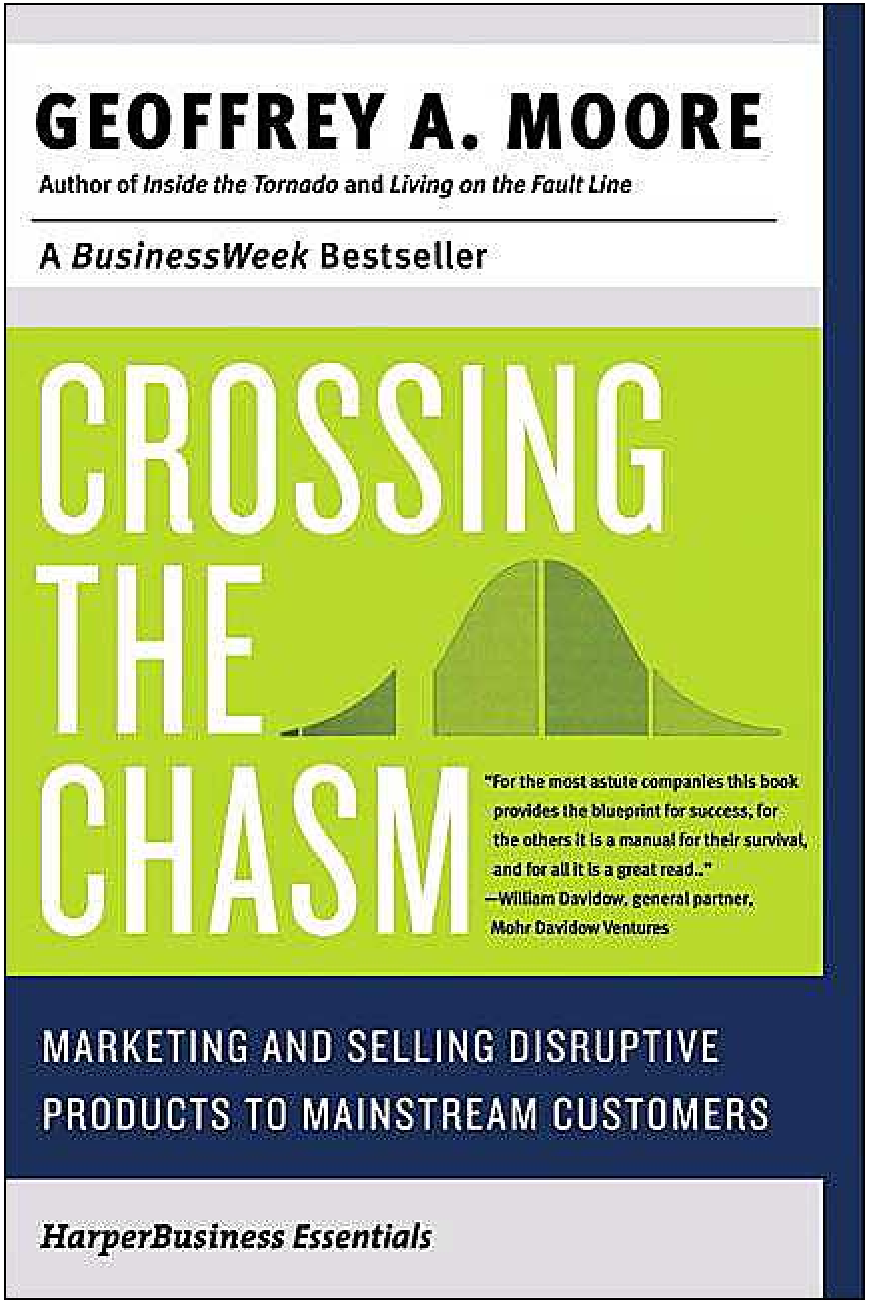
# In Large Companies





# The Von Neumann Bottleneck





GEOFFREY A. MOORE

Author of *Inside the Tornado* and *Living on the Fault Line*

A *BusinessWeek* Bestseller

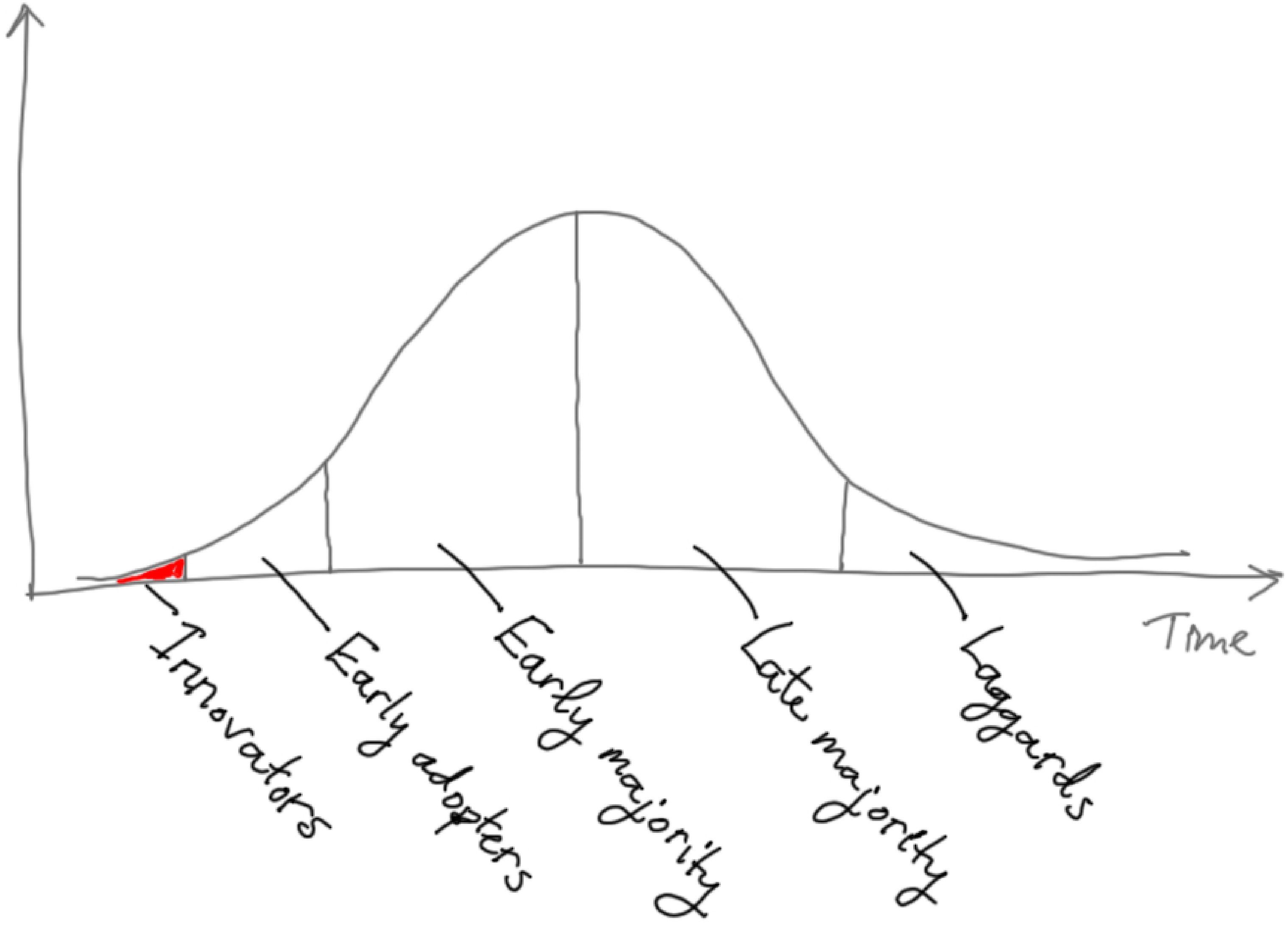
CROSSING THE CHASM

"For the most astute companies this book provides the blueprint for success, for the others it is a manual for their survival, and for all it is a great read.."
—William Davidow, general partner, Mohr Davidow Ventures

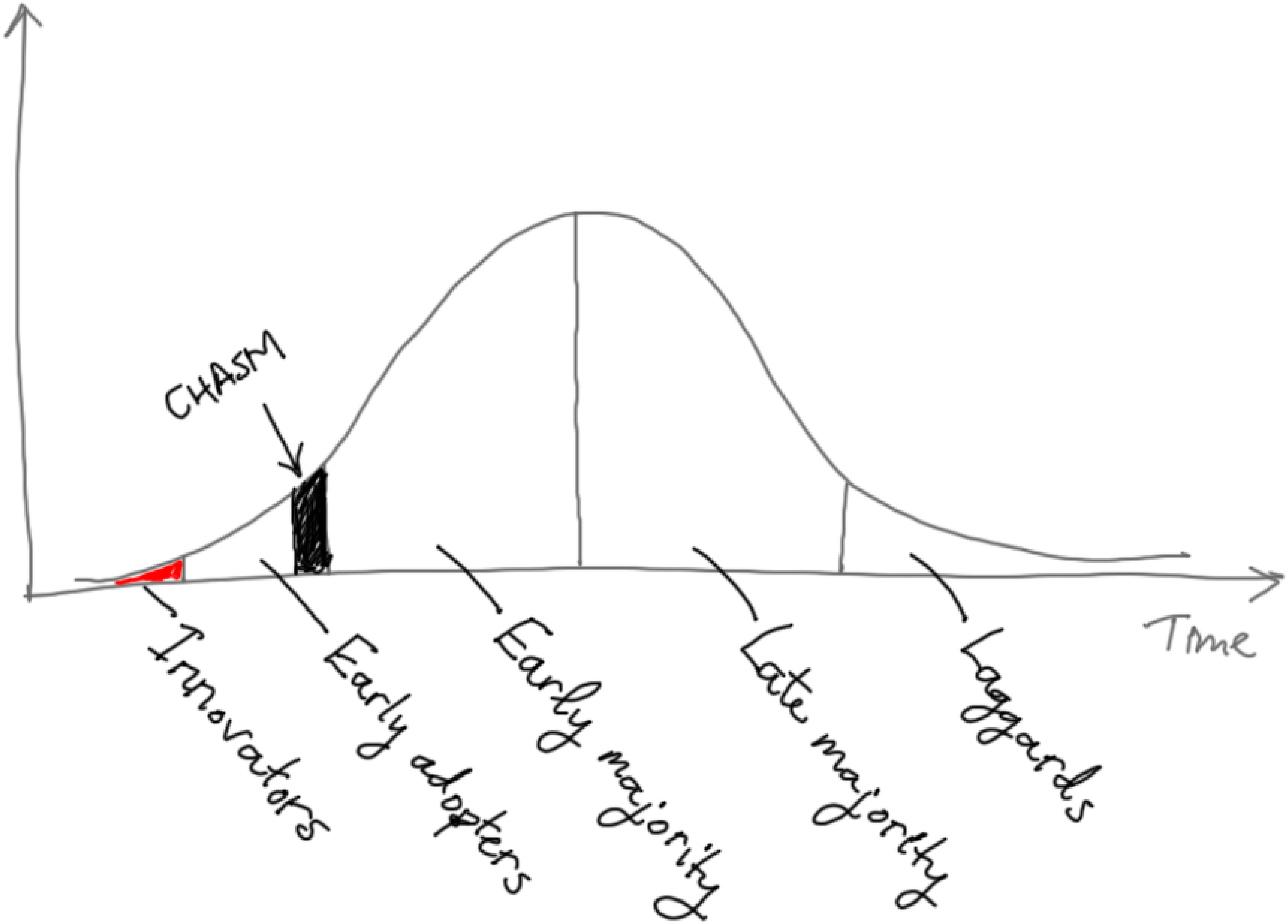
MARKETING AND SELLING DISRUPTIVE PRODUCTS TO MAINSTREAM CUSTOMERS

*HarperBusiness Essentials*

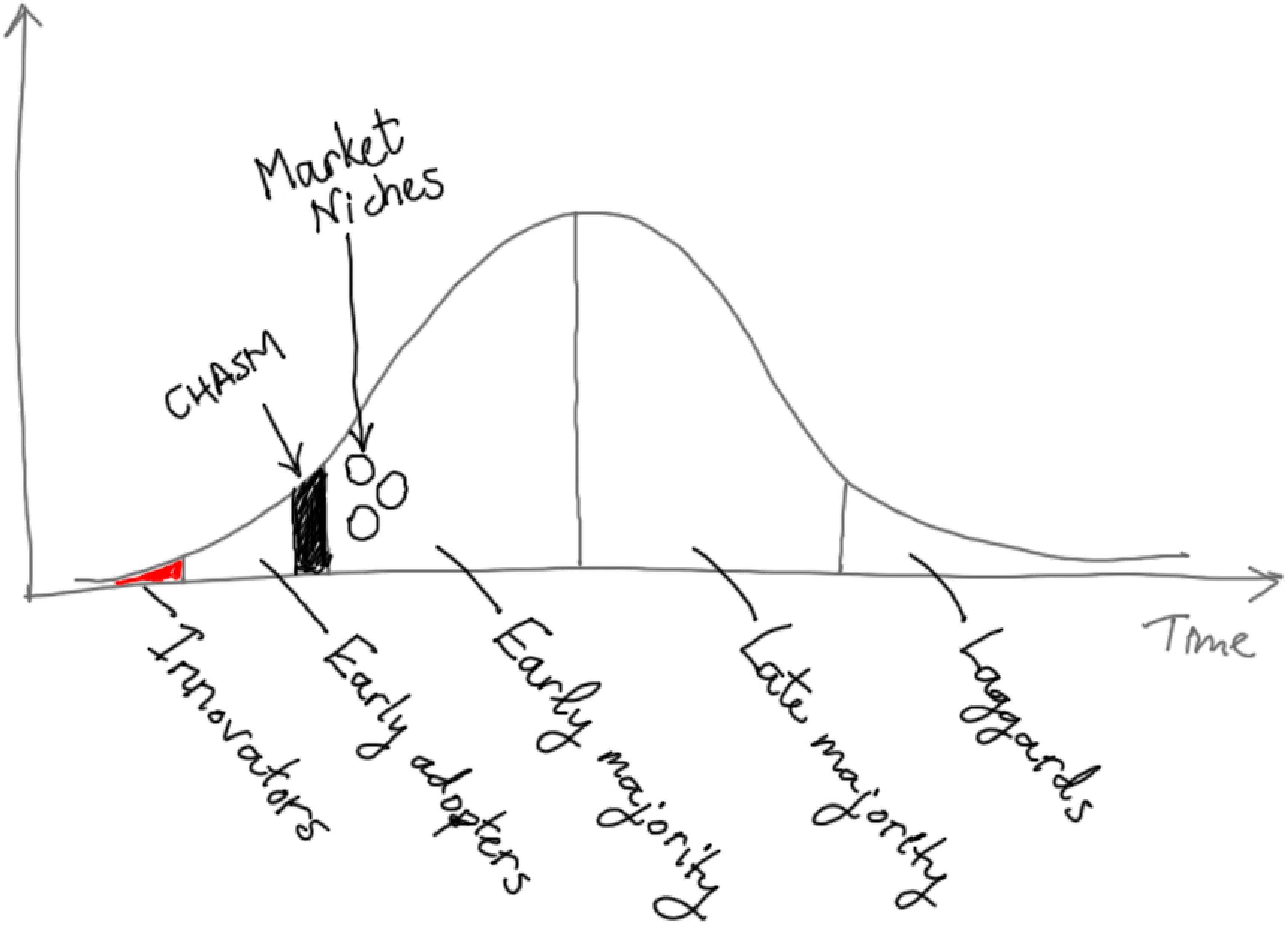
# Technology adoption life cycle



# Technology adoption life cycle



# Technology adoption life cycle



# The Erlang Story

- 1986—Erlang emerges at Ericsson
  - Functional language
  - Extra support for concurrency & fault tolerance
- Early 1990s—small products
- 1996
  - Open Telecoms Platform (higher-order functions for robust telecom systems)
  - AXD 301 project starts

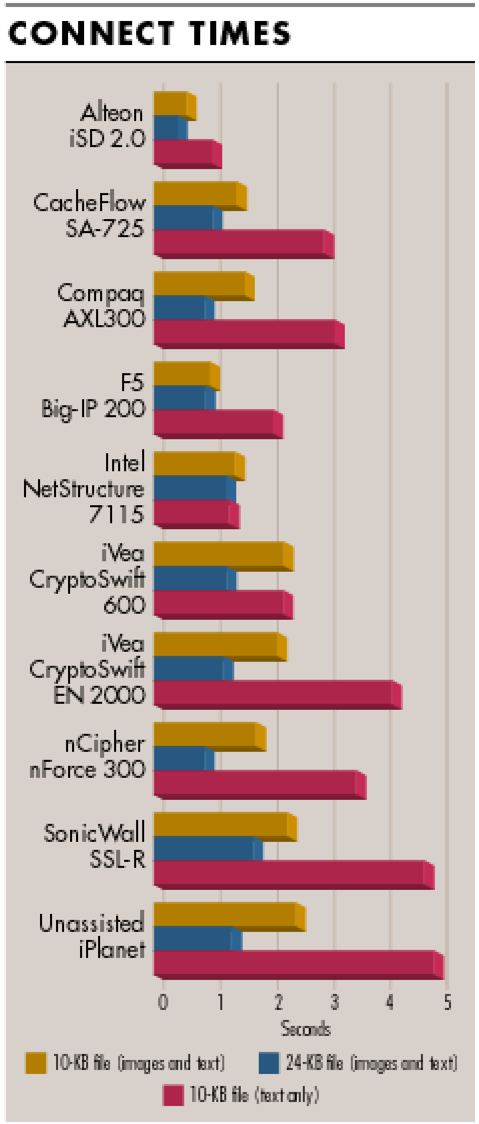
# The AXD 301

- ATM switch (telephone backbone)
- Born out of a failed project!
- 1,5 MLOC Erlang
- *Seven nines* reliability
- 4-10x better productivity, quality

# Erlang Story II

- 1998—Erlang banned for new projects
- 1998—Open source Erlang
- 1998—Bluetail
  - Jane Walerud VD
  - Mail robustifier, Web prioritizer

# SSL Accelerator



- Alteon WebSystems' SSL Accelerator offers phenomenal performance, management and scalability.
  – *Network Computing*

NOW:
2000

“Six years have passed and we hav grown to an ambitious and creative company of 500 employees.”

invoice

“We have been awarded Com of the Year in -07, increased our tur by 13 570 % and become the Nordi market leader.”

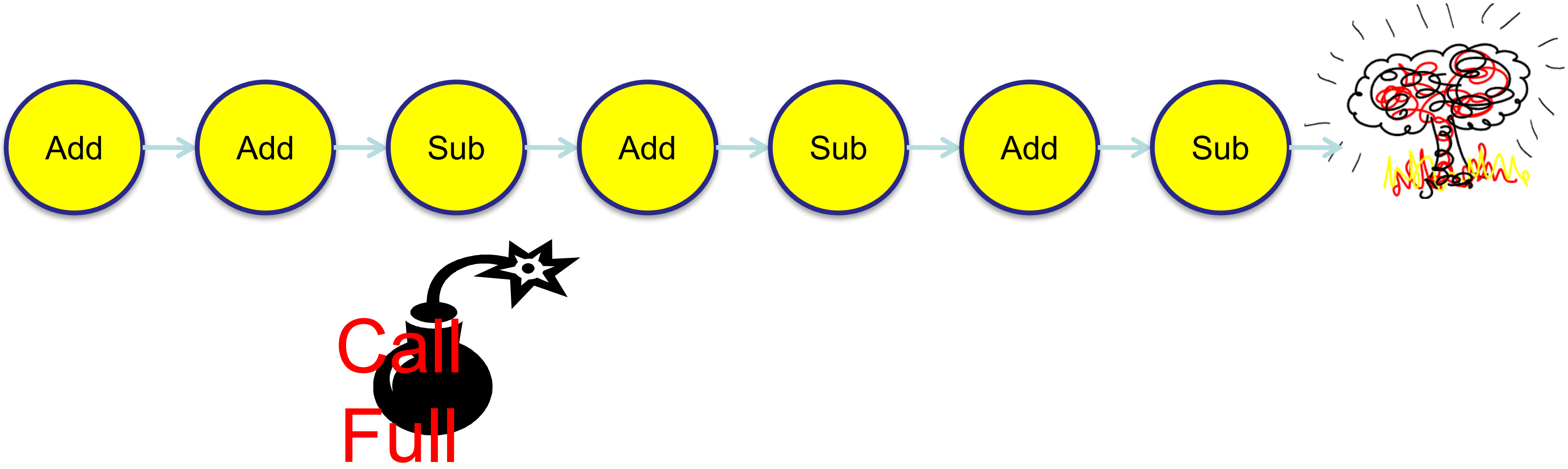
NOW:
in 14
countries

# QuviQ

- Founded May 2006
- Selling… QuickCheck!
  - Key features:
    - Simplifies failing tests
    - Extensions for testing stateful systems
- Testing is a huge problem…

# Media Proxy

- Multimedia IP-telephony (IMS)
- Connects calls across a firewall
- Test adding and removing callers from a call



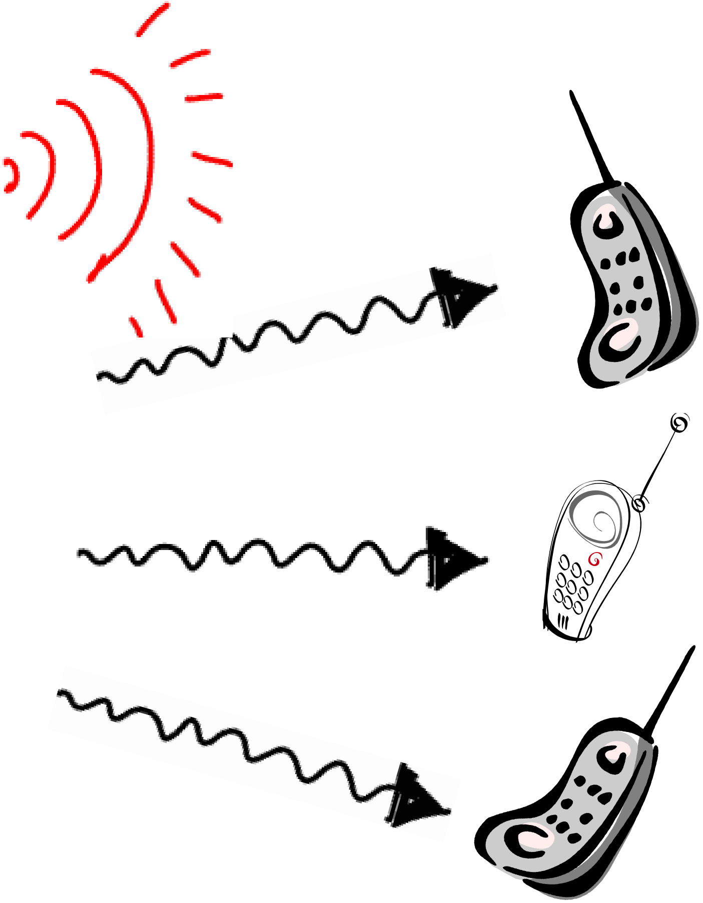
# 3G Radio Base Station

Setup

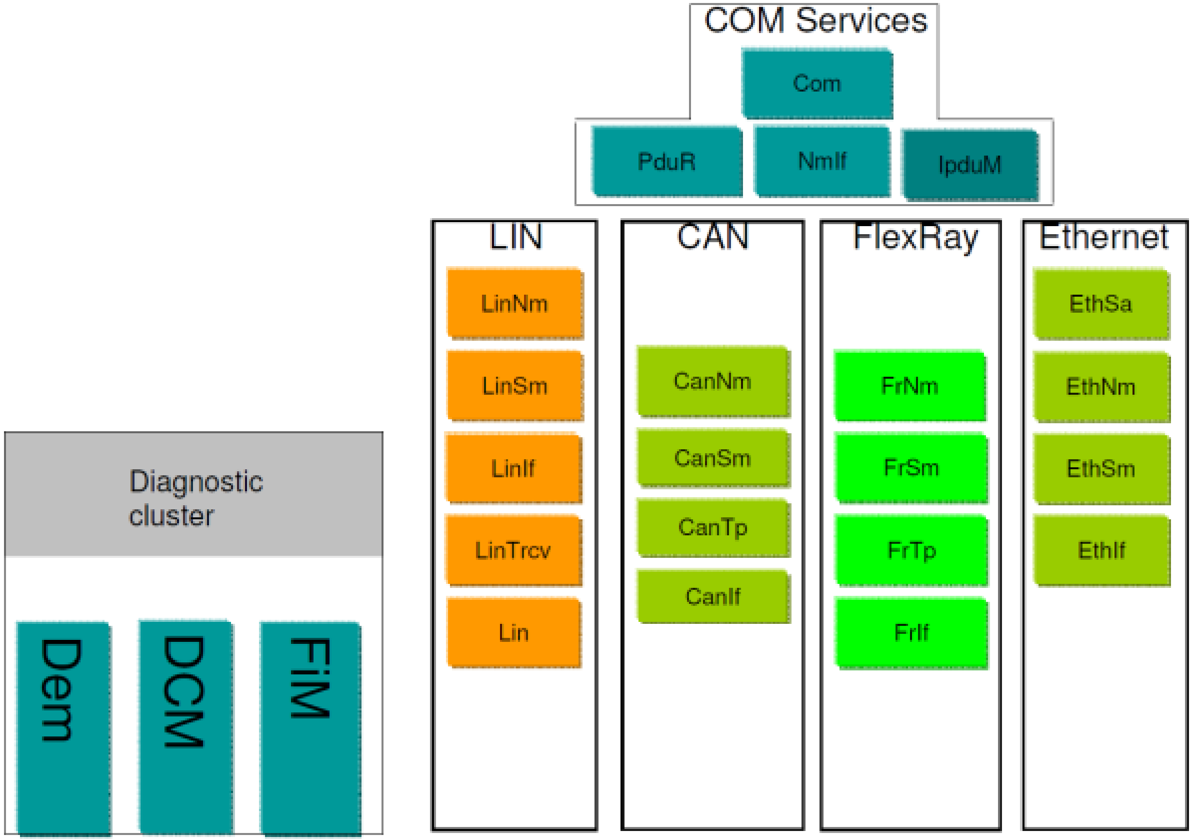
OK

Setup

Reject



Parts of the software tested by QuickCheck



COM Services

Com

PduR

NmIf

IpduM

LIN

LinNm

LinSm

LinIf

LinTrcv

Lin

CAN

CanNm

CanSm

CanTp

CanIf

FlexRay

FrNm

FrSm

FrTp

FrIf

Ethernet

EthSa

EthNm

EthSm

EthIf

Diagnostic cluster

Dem

DCM

FiM

3,000 pages of specifications

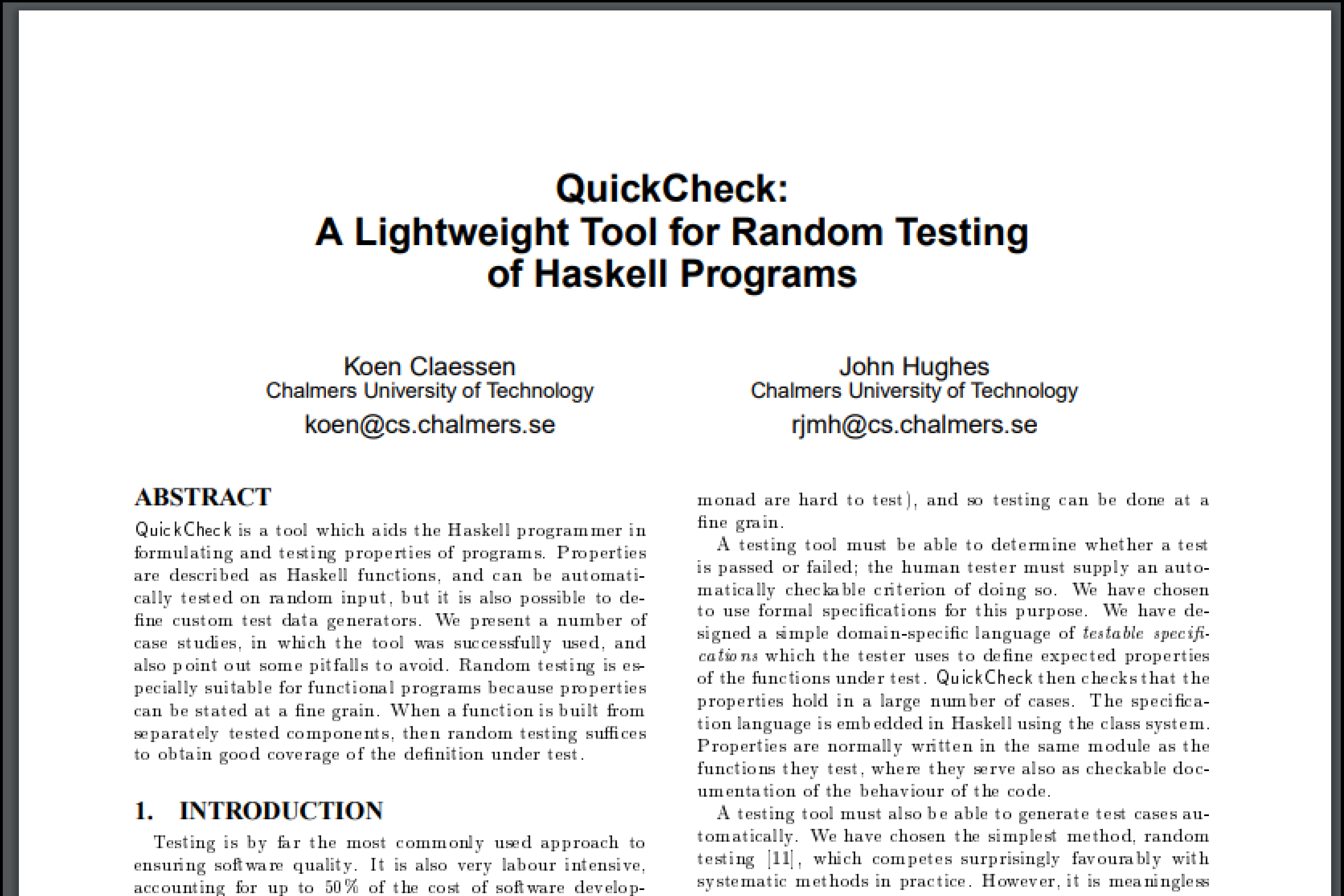
20,000 lines of QuickCheck

1,000,000 LOC, 6 suppliers

200 problems

100 problems in the standard

# ICFP 2000...



## QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs

Koen Claessen
Chalmers University of Technology
koen@cs.chalmers.se

John Hughes
Chalmers University of Technology
rjmh@cs.chalmers.se

### ABSTRACT

QuickCheck is a tool which aids the Haskell programmer in formulating and testing properties of programs. Properties are described as Haskell functions, and can be automatically tested on random input, but it is also possible to define custom test data generators. We present a number of case studies, in which the tool was successfully used, and also point out some pitfalls to avoid. Random testing is especially suitable for functional programs because properties can be stated at a fine grain. When a function is built from separately tested components, then random testing suffices to obtain good coverage of the definition under test.

### 1. INTRODUCTION

Testing is by far the most commonly used approach to ensuring software quality. It is also very labour intensive, accounting for up to 50% of the cost of software develop-

monad are hard to test), and so testing can be done at a fine grain.

A testing tool must be able to determine whether a test is passed or failed; the human tester must supply an automatically checkable criterion of doing so. We have chosen to use formal specifications for this purpose. We have designed a simple domain-specific language of *testable specifications* which the tester uses to define expected properties of the functions under test. QuickCheck then checks that the properties hold in a large number of cases. The specification language is embedded in Haskell using the class system. Properties are normally written in the same module as the functions they test, where they serve also as checkable documentation of the behaviour of the code.

A testing tool must also be able to generate test cases automatically. We have chosen the simplest method, random testing [11], which competes surprisingly favourably with systematic methods in practice. However, it is meaningless

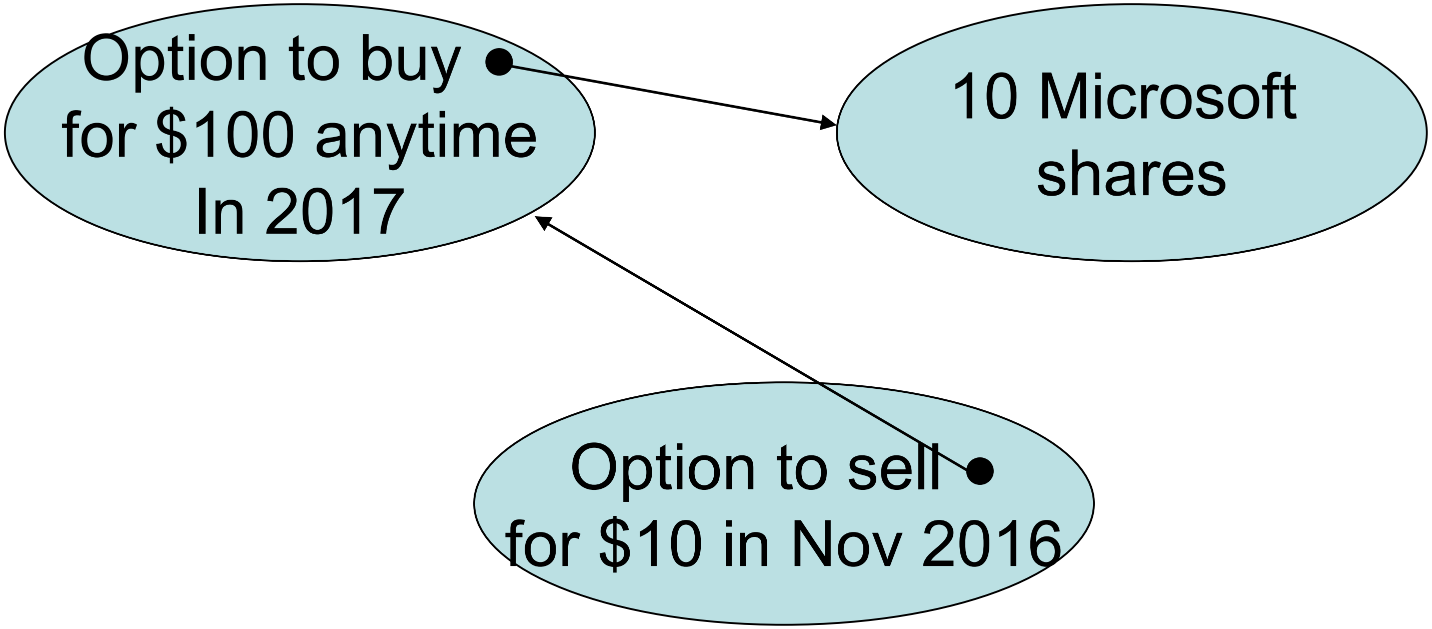
# Most Influential ICFP Paper Award

# Erlang in Ericsson

- 1998—BANNED!
- 2007—Recommended for ”complex state machines with high performance requirements”
- 2010—recruiting Erlang programmers in Göteborg

CREDIT SUISSE

- Derivatives trading in New York

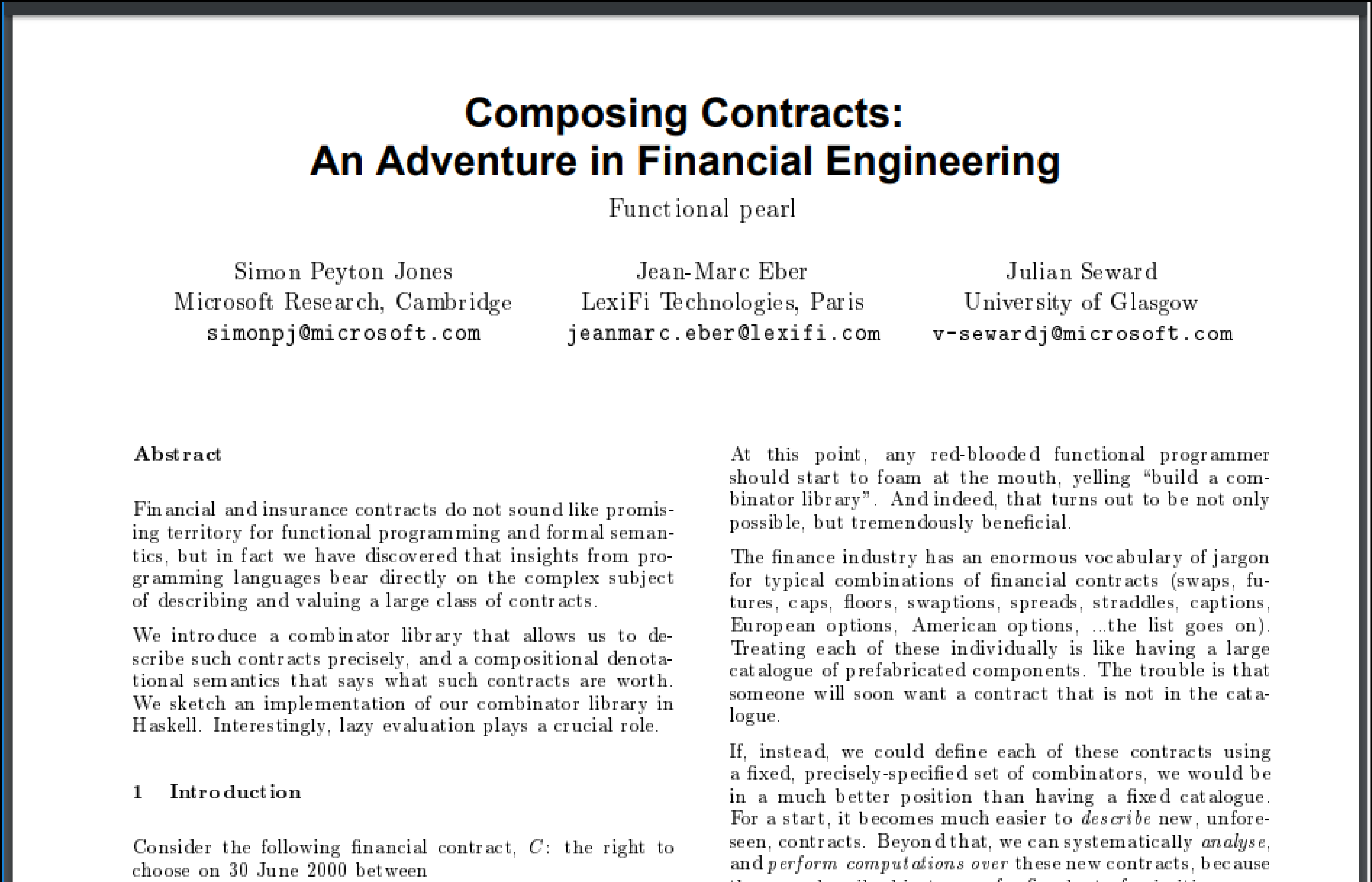


Option to buy for $100 anytime In 2017

10 Microsoft shares

Option to sell for $10 in Nov 2016

# Also in ICFP 2000...



## Composing Contracts: An Adventure in Financial Engineering

Functional pearl

Simon Peyton Jones
Microsoft Research, Cambridge
simonpj@microsoft.com

Jean-Marc Eber
LexiFi Technologies, Paris
jeanmarc.eber@lexifi.com

Julian Seward
University of Glasgow
v-sewardj@microsoft.com

### Abstract

Financial and insurance contracts do not sound like promising territory for functional programming and formal semantics, but in fact we have discovered that insights from programming languages bear directly on the complex subject of describing and valuing a large class of contracts.

We introduce a combinator library that allows us to describe such contracts precisely, and a compositional denotational semantics that says what such contracts are worth. We sketch an implementation of our combinator library in Haskell. Interestingly, lazy evaluation plays a crucial role.

### 1 Introduction

Consider the following financial contract, $C$: the right to choose on 30 June 2000 between

At this point, any red-blooded functional programmer should start to foam at the mouth, yelling "build a combinator library". And indeed, that turns out to be not only possible, but tremendously beneficial.

The finance industry has an enormous vocabulary of jargon for typical combinations of financial contracts (swaps, futures, caps, floors, swaptions, spreads, straddles, captions, European options, American options, ...the list goes on). Treating each of these individually is like having a large catalogue of prefabricated components. The trouble is that someone will soon want a contract that is not in the catalogue.

If, instead, we could define each of these contracts using a fixed, precisely-specified set of combinators, we would be in a much better position than having a fixed catalogue. For a start, it becomes much easier to *describe* new, unforeseen, contracts. Beyond that, we can systematically *analyse*, and *perform computations over* these new contracts, because

# Financial Contracts in Haskell

- The option to acquire 10 Microsoft shares, for $100, anytime

```
anytime :: Contract -> Contract
-- Acquire the underlying contract at
-- any time before it expires (but
-- you must acquire it)
```

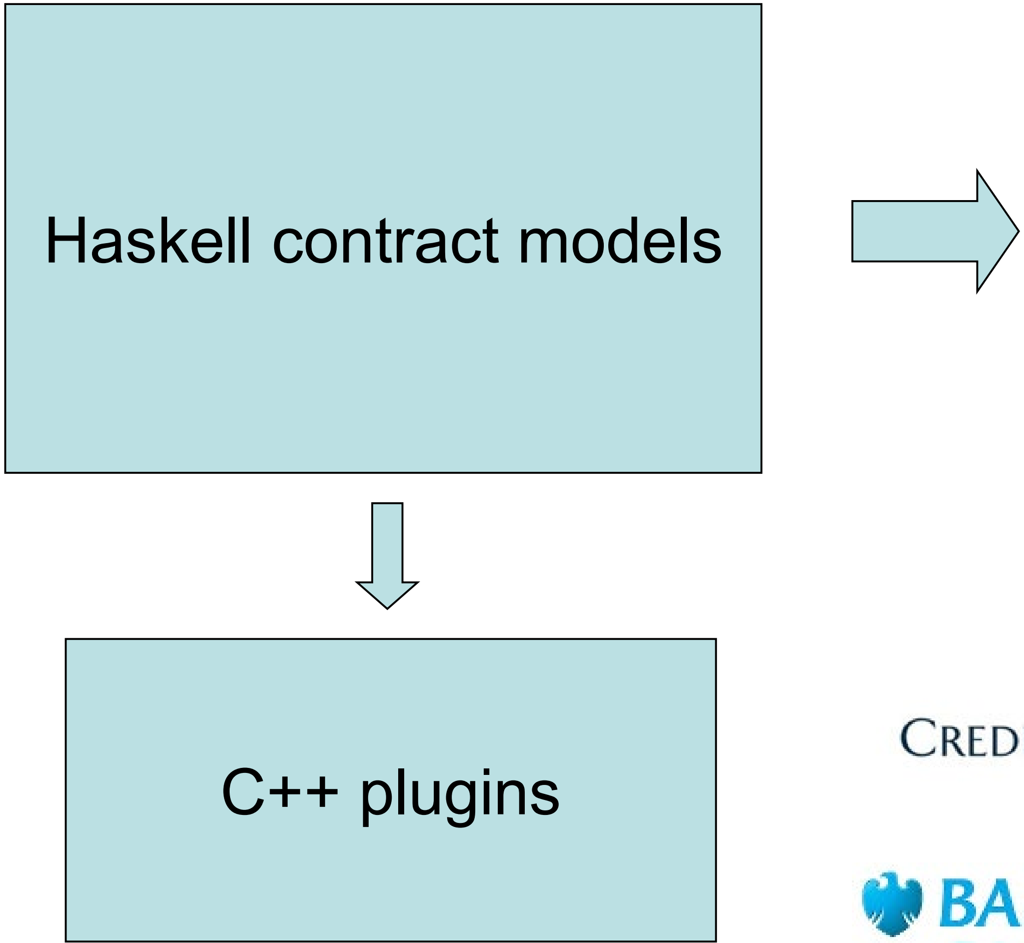
**anytime**: Choose *when*

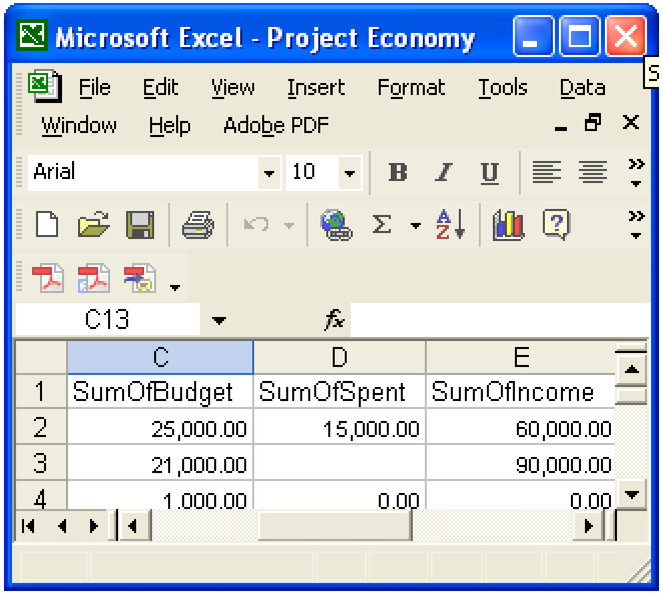
```
golden_handcuff = anytime shares

shares = zero `or` (scaleK -100 (one Dollar) `and`
                    scaleK 10 (one MSShare))
```

**or**: Choose *whether*

MS shares are a “currency”

# New Approach




Microsoft Excel - Project Economy
File Edit View Insert Format Tools Data
Window Help Adobe PDF
Arial 10
B I U
fx

C13

|   | C | D | E |
|---|---|---|---|
| 1 | SumOfBudget | SumOfSpent | SumOfIncome |
| 2 | 25,000.00 | 15,000.00 | 60,000.00 |
| 3 | 21,000.00 |  | 90,000.00 |
| 4 | 1,000.00 | 0.00 | 0.00 |

CREDIT SUISSE

Standard Chartered

BARCLAYS CAPITAL

Bloomberg

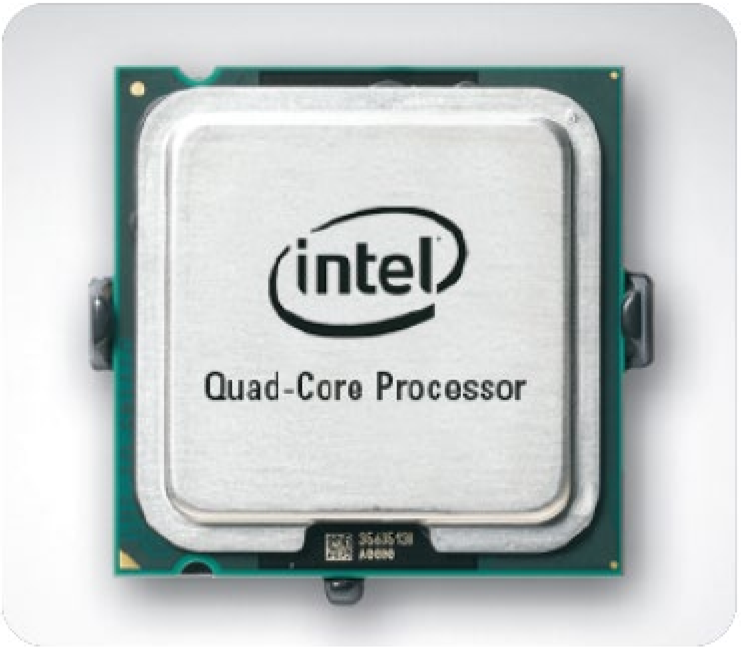
Goldman Sachs

JANE ST. CAPITAL

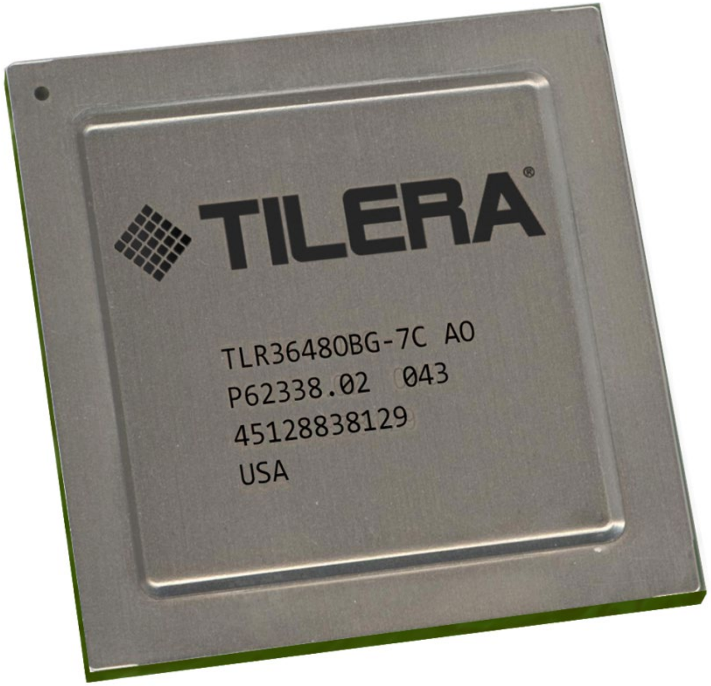
- ”Functional programming on Wall Street”
  - Proprietary trading, ~$13 billion/day
  - >400 people, in New York, London, Hong Kong, Amsterdam

  - OCaml primary development language
  - Hire summer interns every year

# The Multicore Opportunity



4 cores

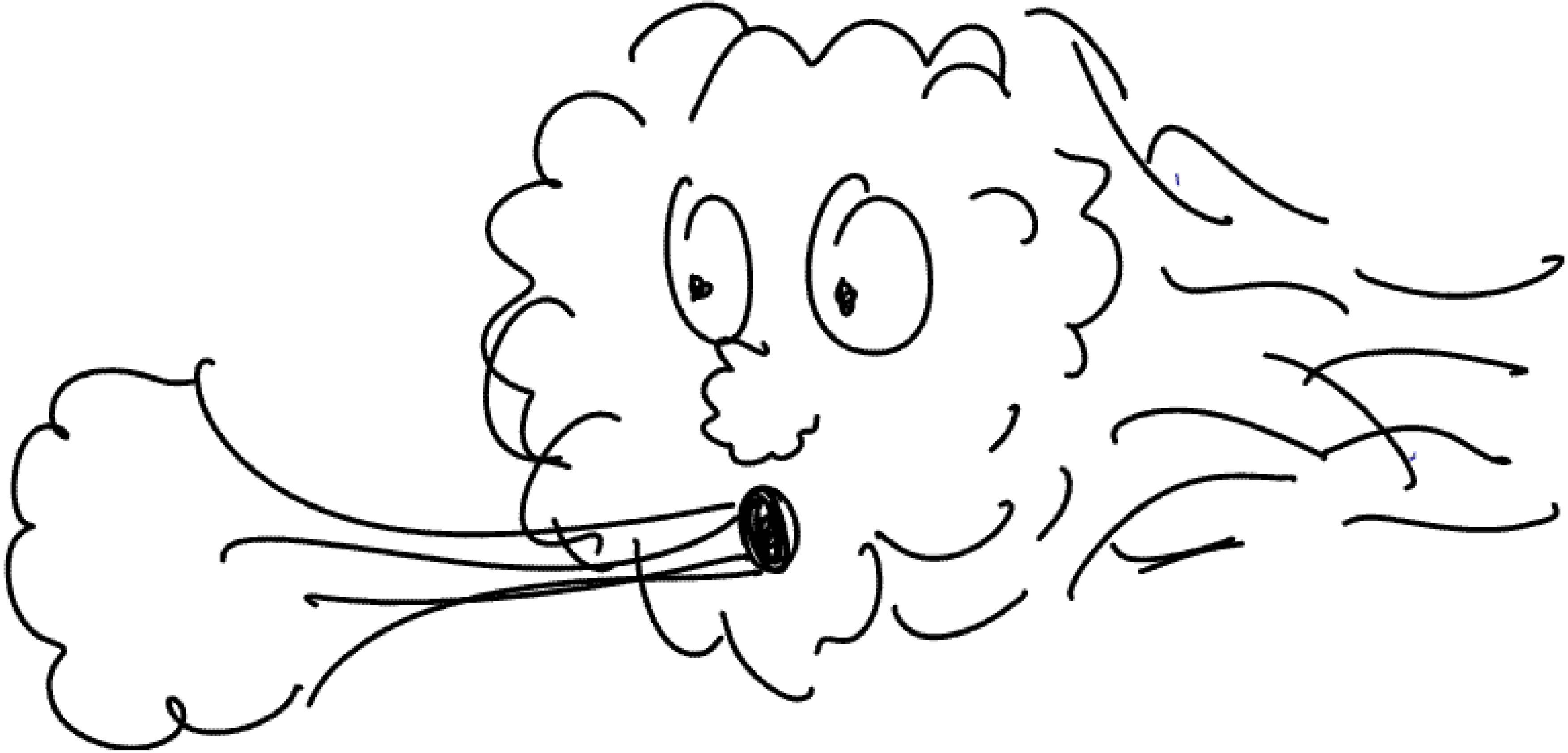


100 cores

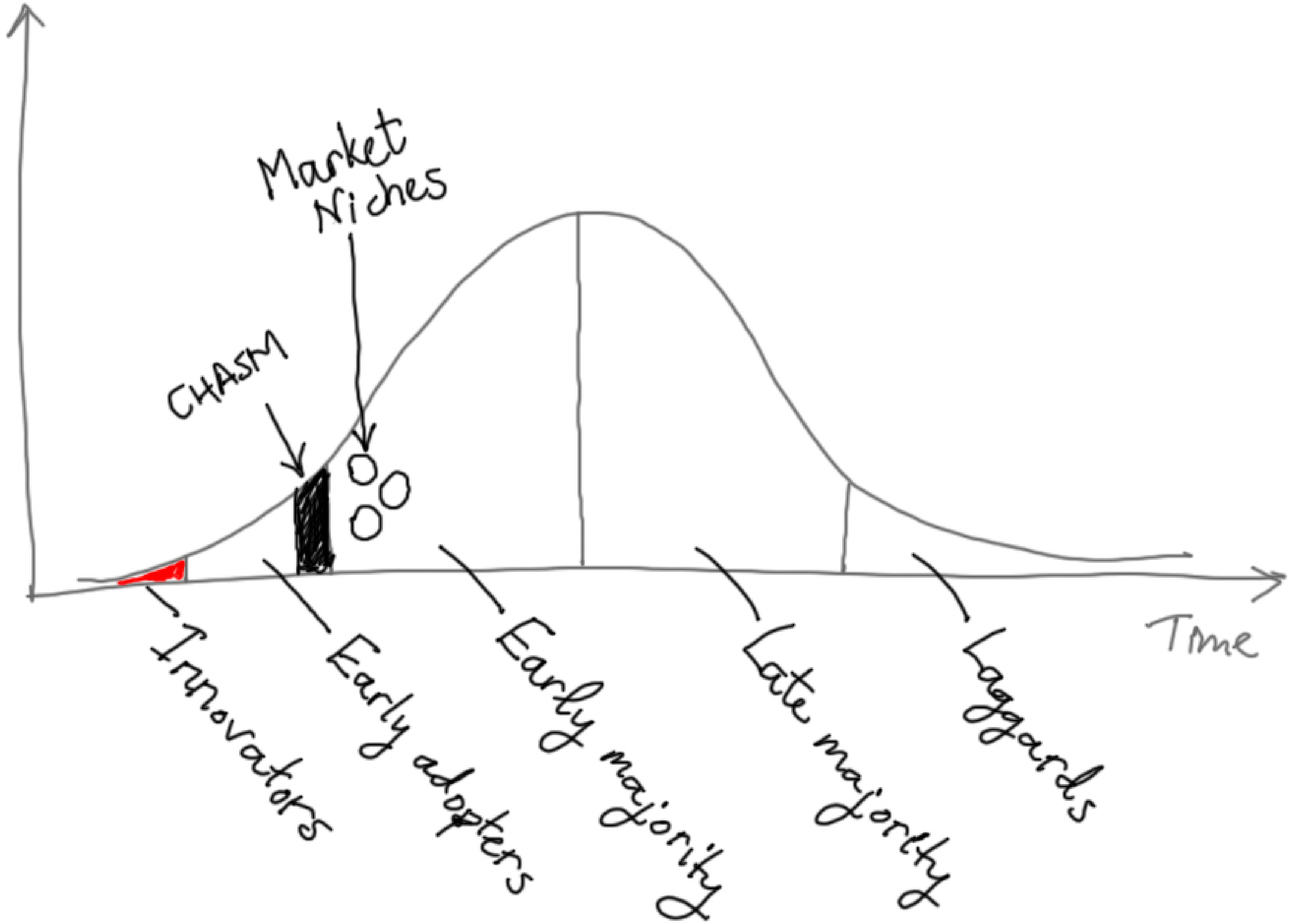
# Intel CTO Justin Rattner

Sept 15, 2011, EETimes, keynote speech at Intel Developers' Forum

- “The new software represents an effort to bring to today's C++ programmers some of the concepts of the emerging school called functional programming. Functional programming looks to be one of the foundations for parallel progra going forward with higher l abstraction and mor n of parallelism.”



# Where are we?


Market
Niches
CHASM
Time
Innovators
Early adopters
Early majority
Late majority
Laggards

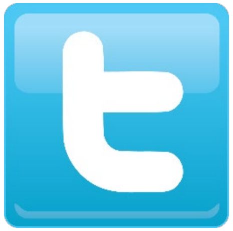
# Some popular functional languages…

# Some languages with lambda expressions

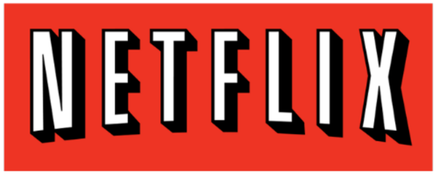
- C# — since 3.0 (2008)
- C++ — since C++11
- Rust
- Java — since Java 8 (2014)

- Visual Basic — since version 9 (2008)

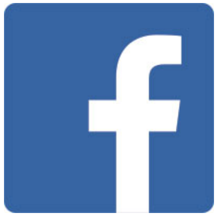
# Some high-profile applications



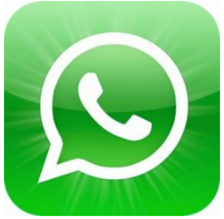
Servers programmed in Scala



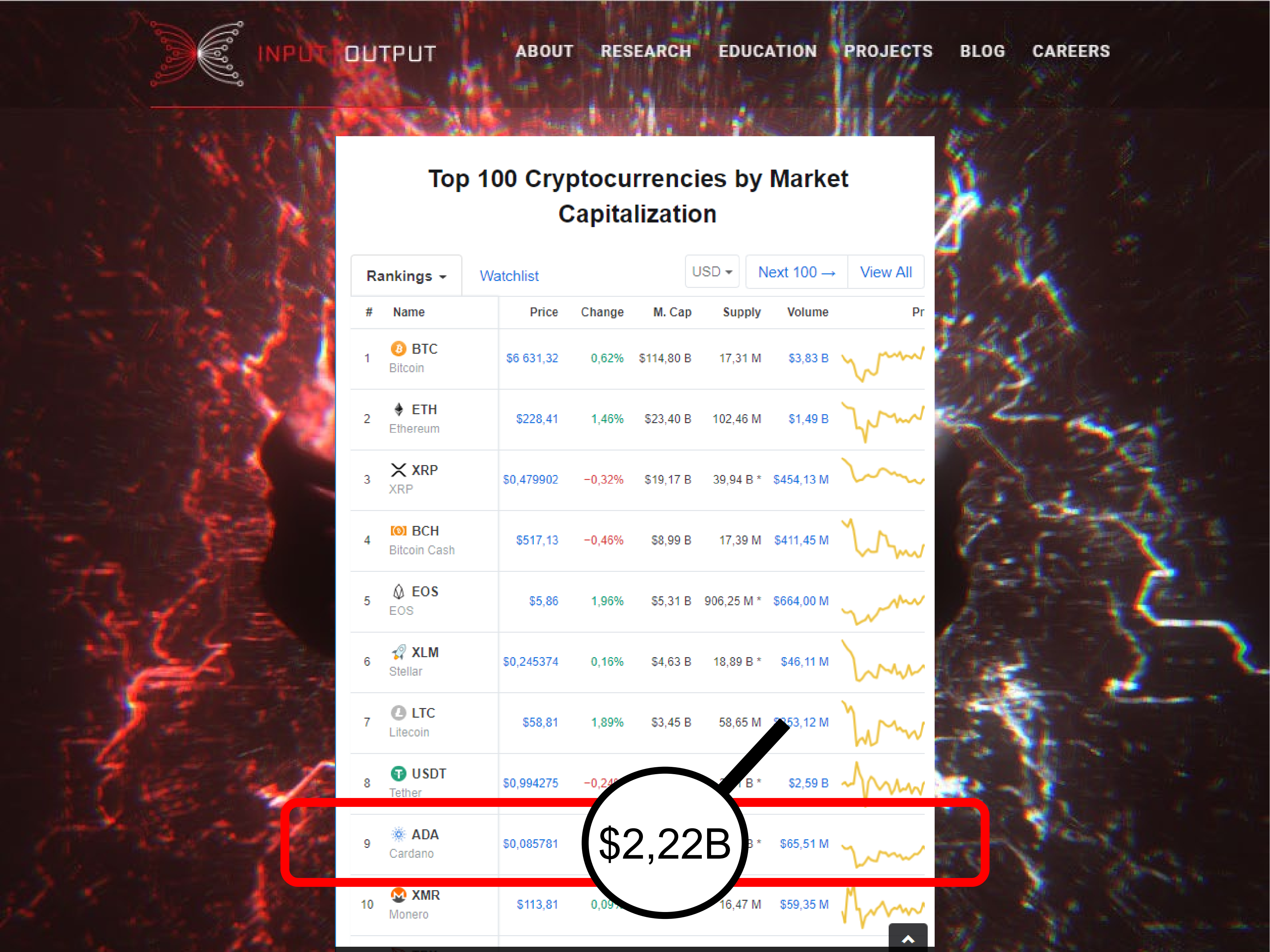
”Functional reactive programming” (with RxJava)



Spam detection in Haskell



Servers in Erlang



## Top 100 Cryptocurrencies by Market Capitalization

| # | Name | Price | Change | M. Cap | Supply | Volume | Pr |
|---|---|---|---|---|---|---|---|
| 1 | BTC<br>Bitcoin | $6 631,32 | 0,62% | $114,80 B | 17,31 M | $3,83 B | |
| 2 | ETH<br>Ethereum | $228,41 | 1,46% | $23,40 B | 102,46 M | $1,49 B | |
| 3 | XRP<br>XRP | $0,479902 | −0,32% | $19,17 B | 39,94 B * | $454,13 M | |
| 4 | BCH<br>Bitcoin Cash | $517,13 | −0,46% | $8,99 B | 17,39 M | $411,45 M | |
| 5 | EOS<br>EOS | $5,86 | 1,96% | $5,31 B | 906,25 M * | $664,00 M | |
| 6 | XLM<br>Stellar | $0,245374 | 0,16% | $4,63 B | 18,89 B * | $46,11 M | |
| 7 | LTC<br>Litecoin | $58,81 | 1,89% | $3,45 B | 58,65 M | [illegible]53,12 M | |
| 8 | USDT<br>Tether | $0,994275 | −0,24 | [illegible] | [illegible] B * | $2,59 B | |
| 9 | ADA<br>Cardano | $0,085781 | [illegible] | [illegible] | [illegible] B * | $65,51 M | |
| 10 | XMR<br>Monero | $113,81 | 0,09 | [illegible] | 16,47 M | $59,35 M | |

$2,22B

# Some recent success stories

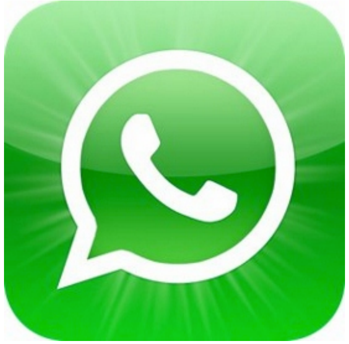
Swedish start-up using Erlang, sold to Cisco for $175 million in 2014

New Jersey e-commerce start-up using F#, sold to Walmart for $3.3 *billion* in 2016

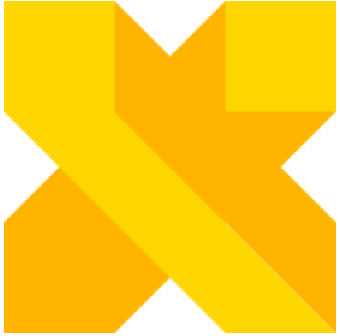
“So, we started building two solutions, a C# solution and an F# solution, to see where they would take us. In the end, we chose to stick with the F# path. The main reason is that we could deliver the same functionality with far less code. This clearly eases maintainability and reduces bugs.”

*—Marie-France Han, Jet tech blog*

# Some recent success stories



Bay area start-up using Erlang, sold to Facebook for $22 *billion* in 2014



???

ICFP
2018

566 attendees

>45% from industry

ahrefs

Jane Street

MIRI
MACHINE INTELLIGENCE
RESEARCH INSTITUTE

Standard Chartered

DFINITY

facebook

moz://a

Strange Loop

SEPTEMBER 12-14 2019 / STIFEL THEATRE / ST. LOUIS, MO

Meet us in St. Louis, Sept 12-14th, 2019, to make connections with the creators and users of the languages, libraries, tools, and techniques at the forefront of the industry.

Developer conference

2,200 attendees!

Erlang FACTORY

London,
Stockholm,
San Francisco

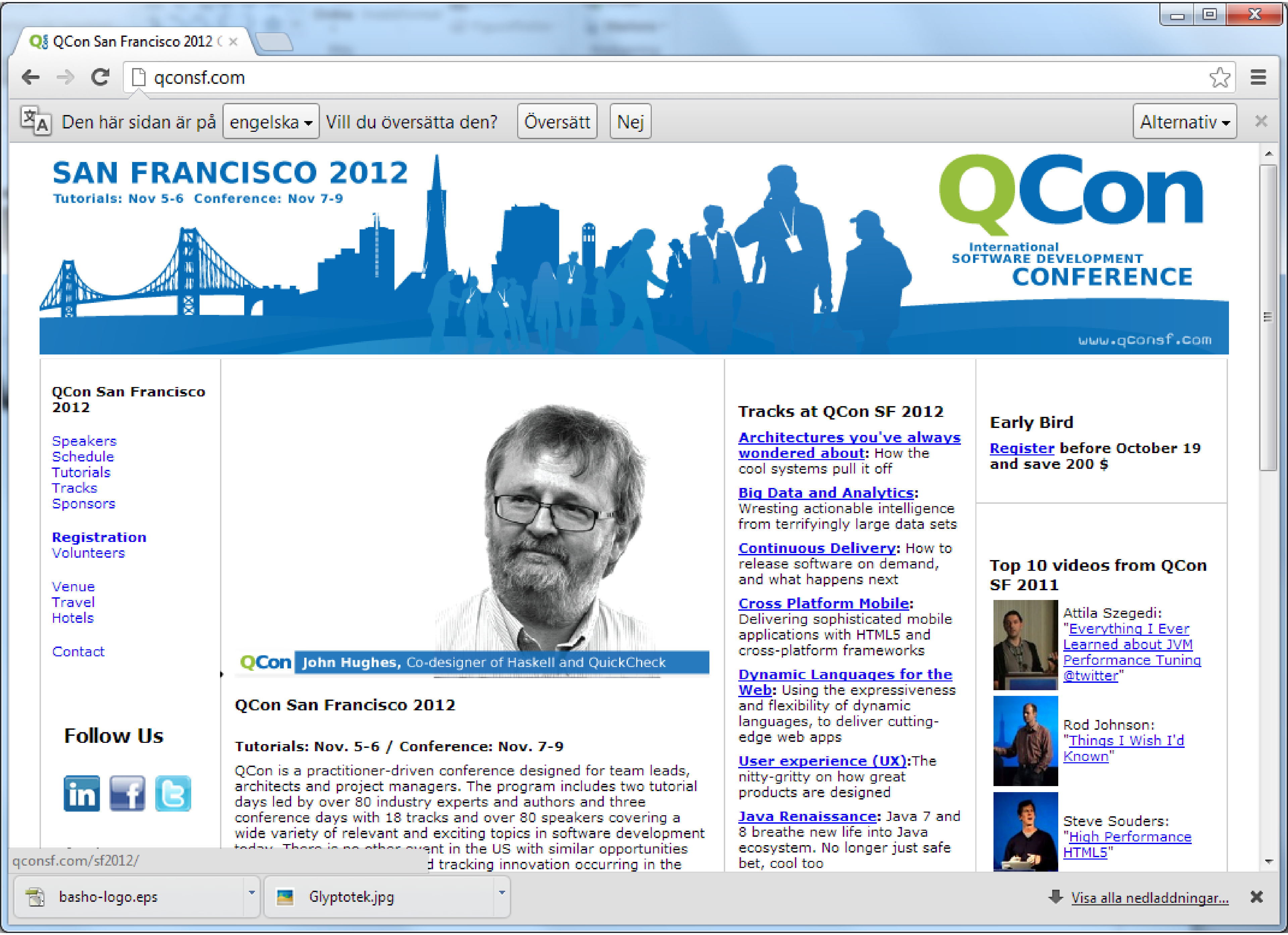
New York,
Berlin

lambda DAλS

Krakow

YOW!
LAMBDA JAM

Sydney


QCon San Francisco 2012 (
qconsf.com
Den här sidan är på engelska Vill du översätta den? Översätt Nej Alternativ
SAN FRANCISCO 2012
Tutorials: Nov 5-6 Conference: Nov 7-9
QCon
International SOFTWARE DEVELOPMENT CONFERENCE
www.qconsf.com
QCon John Hughes, Co-designer of Haskell and QuickCheck
qconsf.com/sf2012/
basho-logo.eps
Glyptotek.jpg
Visa alla nedladdningar...

QCon San Francisco 2012

Speakers
Schedule
Tutorials
Tracks
Sponsors

**Registration**
Volunteers

Venue
Travel
Hotels

Contact

**Follow Us**

**QCon San Francisco 2012**

**Tutorials: Nov. 5-6 / Conference: Nov. 7-9**

QCon is a practitioner-driven conference designed for team leads, architects and project managers. The program includes two tutorial days led by over 80 industry experts and authors and three conference days with 18 tracks and over 80 speakers covering a wide variety of relevant and exciting topics in software development today. There is no other event in the US with similar opportunities … d tracking innovation occurring in the

**Tracks at QCon SF 2012**

**Architectures you've always wondered about**: How the cool systems pull it off

**Big Data and Analytics**: Wresting actionable intelligence from terrifyingly large data sets

**Continuous Delivery**: How to release software on demand, and what happens next

**Cross Platform Mobile**: Delivering sophisticated mobile applications with HTML5 and cross-platform frameworks

**Dynamic Languages for the Web**: Using the expressiveness and flexibility of dynamic languages, to deliver cutting-edge web apps

**User experience (UX)**:The nitty-gritty on how great products are designed

**Java Renaissance**: Java 7 and 8 breathe new life into Java ecosystem. No longer just safe bet, cool too

**Early Bird**

**Register before October 19 and save 200 $**

**Top 10 videos from QCon SF 2011**

Attila Szegedi: "Everything I Ever Learned about JVM Performance Tuning @twitter"

Rod Johnson: "Things I Wish I'd Known"

Steve Souders: "High Performance HTML5"

# If you'd like to see more…

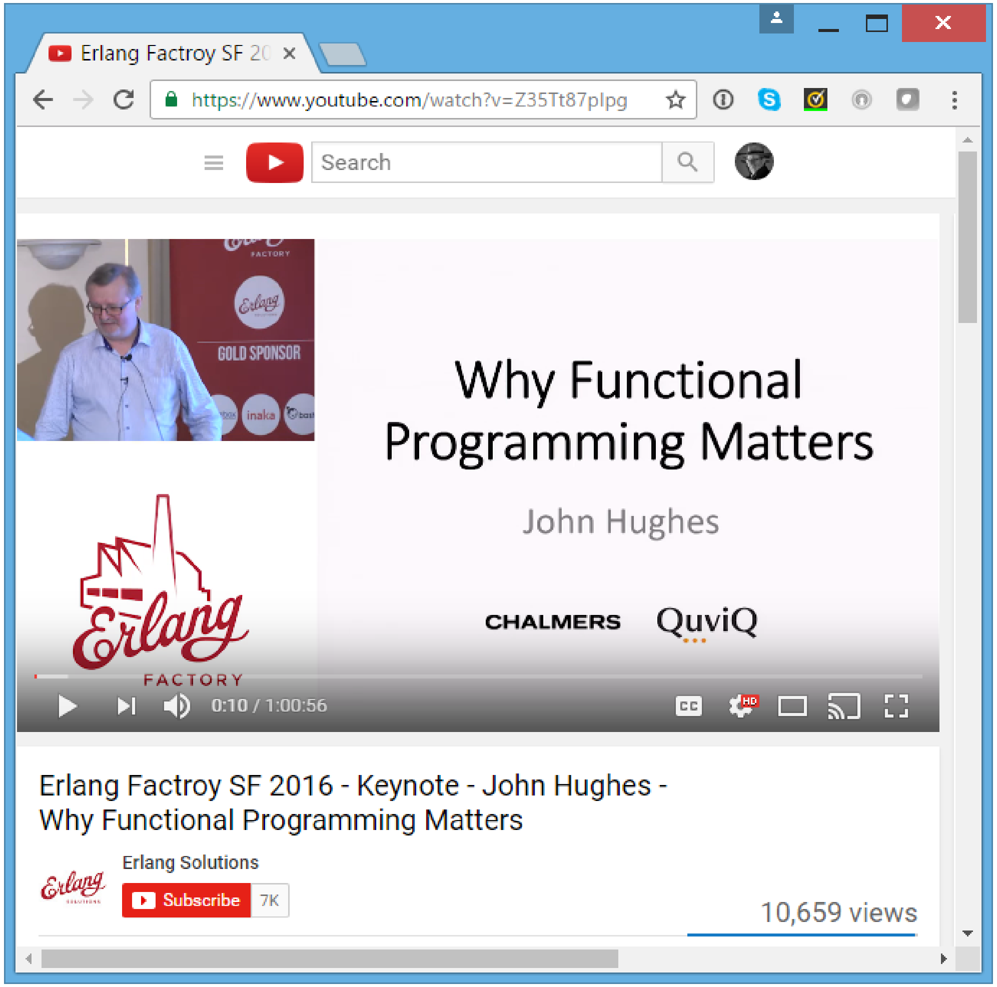
Google:

**John Hughes Keynote**

It's the first hit.

(San Francisco 2016 or Krakow 2017)

1 hour: watch when you have time



Erlang Factroy SF 2016 - Keynote - John Hughes - Why Functional Programming Matters