What has mathematics to do with software engineering?

Perdita Stevens
Professor of Mathematics of Software Engineering

University of Edinburgh

February 2016

1966

I was born.

By 1968, there was a software crisis.

What is software? What is a software crisis?

Software: the parts of a computer you can't break with a hammer.

Software crisis: not (just) about software; not (just) a crisis.

Making good software is slow, difficult and expensive.

The origin of the term "software crisis"

seems to have been the 1968 NATO Conference on Software Engineering.

That was *not* the origin of "software engineering", quite.

Earliest use I know – thanks to Bertrand Meyer – is by Anthony A. Oettinger, then ACM President, in August 1966.

Was the software crisis a mathematicians' conspiracy?

Looking first at the origins of the "software crisis" I note that this specific phrase appears only in editorial material [...] not in any of the quoted dialog between participants.

Thomas Haigh

I thus identify the editors of the volumes as key agents in its original promulgation, but find that computer scientist Edsger Dijkstra was responsible for its more widespread adoption in the 1970s during a quixotic campaign to evict almost all practicing programmers from their jobs and replace them with mathematicians.

`http://tomandmaria.com/Tom/Writing/SoftwareCrisis_SofiaDRAFT.pdf`

Dijkstra quotation from 1968 NATO conference

We, in the Netherlands, have the title Mathematical Engineer. Software engineering seems to be the activity for the Mathematical Engineer par excellence. This seems to fit perfectly. On the one hand, we have all the aspects of an engineering activity, in that you are making something and want to see that it really works. On the other hand, our basic tools are mathematical in nature.

I want to add another question or remark to your list. You are right in saying that lots of systems really work, these are our glimmer of hope. But there is a profound difference between observing that apparently some people are able to do something, and being able to teach that ability.

Mathematical in nature?

[M]athematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true.

Bertrand Russell

Mathematicians are like Frenchmen: whatever you say to them they translate into their own language and forthwith it is something entirely different.

Johann Wolfgang von Goethe

Mathematical in nature?

A mathematician is a machine for turning coffee into theorems.

Alfréd Rényi (via Paul Erdős)

$$M : C \longrightarrow Th$$

Hence

A comathematician is a machine for turning cotheorems into ffee.

$$\overline{M} : \overline{Th} \rightarrow \overline{C}$$

Mathematical in nature?

$$M : C \to Th$$

$$\overrightarrow{M} : C \to Th$$

$$\overrightarrow{M} : C \times Th \to Th$$

$$\overline{M} : \overline{Th} \to \overline{C}$$

$$\overleftarrow{M} : Th \to C$$

$$\overleftarrow{M} : C \times Th \to C$$

$$M(c, th) \Leftrightarrow (cold(c) \wedge false(th)) \vee (hot(c) \wedge true(th))$$

But seriously...

Let's go with:

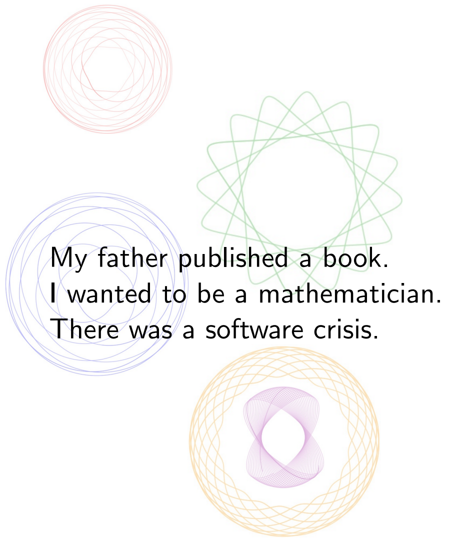
Mathematics is the study of patterns.

just about everyone: "About 26,600 results"

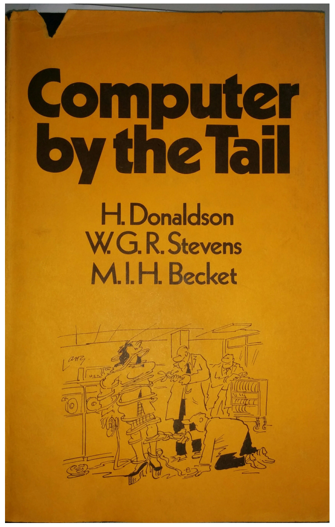
Software is chock full of patterns!

So perhaps: software engineering is a branch of applied mathematics.

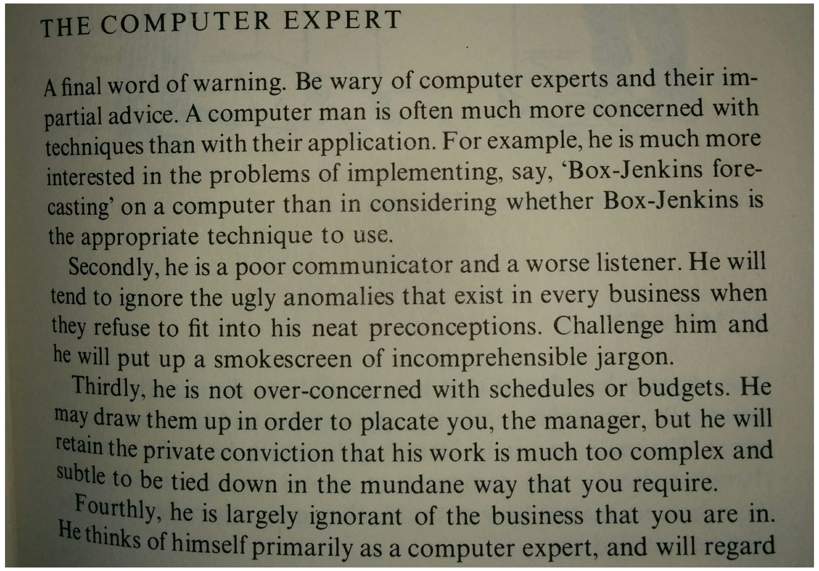
1976



My father published a book.
I wanted to be a mathematician.
There was a software crisis.



Plus ça change



THE COMPUTER EXPERT

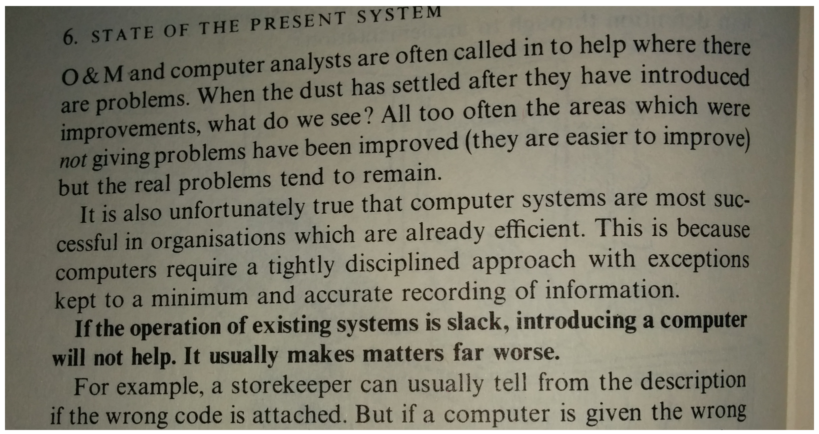
A final word of warning. Be wary of computer experts and their impartial advice. A computer man is often much more concerned with techniques than with their application. For example, he is much more interested in the problems of implementing, say, 'Box-Jenkins forecasting' on a computer than in considering whether Box-Jenkins is the appropriate technique to use.

Secondly, he is a poor communicator and a worse listener. He will tend to ignore the ugly anomalies that exist in every business when they refuse to fit into his neat preconceptions. Challenge him and he will put up a smokescreen of incomprehensible jargon.

Thirdly, he is not over-concerned with schedules or budgets. He may draw them up in order to placate you, the manager, but he will retain the private conviction that his work is much too complex and subtle to be tied down in the mundane way that you require.

Fourthly, he is largely ignorant of the business that you are in. He thinks of himself primarily as a computer expert, and will regard

plus c'est la même chose


6. STATE OF THE PRESENT SYSTEM

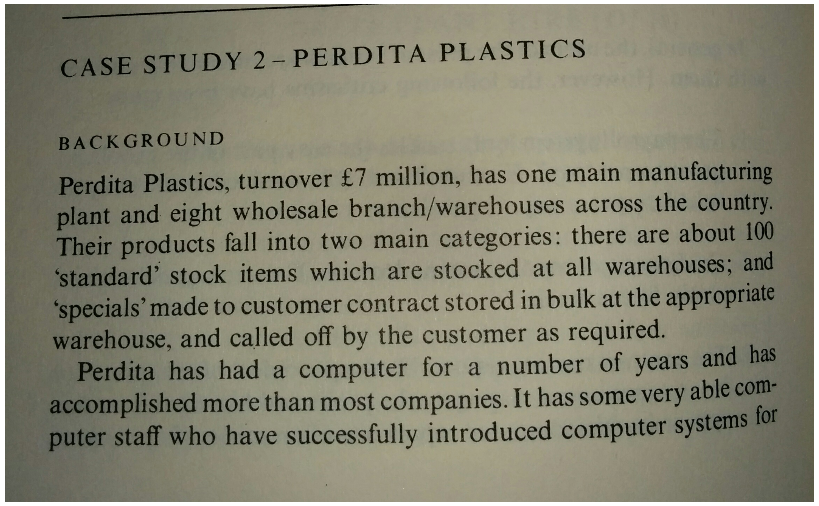
O & M and computer analysts are often called in to help where there are problems. When the dust has settled after they have introduced improvements, what do we see? All too often the areas which were *not* giving problems have been improved (they are easier to improve) but the real problems tend to remain.

It is also unfortunately true that computer systems are most successful in organisations which are already efficient. This is because computers require a tightly disciplined approach with exceptions kept to a minimum and accurate recording of information.

If the operation of existing systems is slack, introducing a computer will not help. It usually makes matters far worse.

For example, a storekeeper can usually tell from the description if the wrong code is attached. But if a computer is given the wrong

but at least it got me mentioned in print

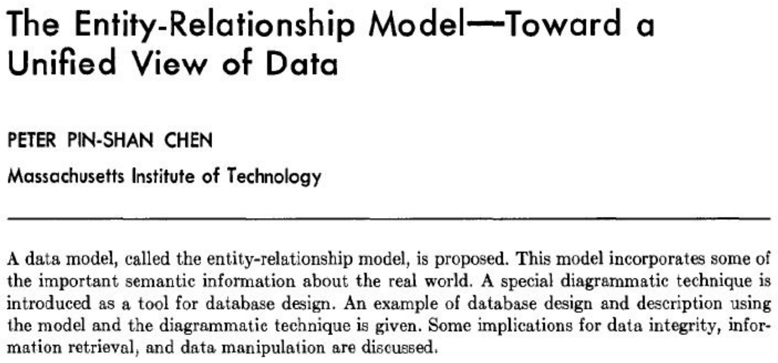

CASE STUDY 2 – PERDITA PLASTICS

BACKGROUND

Perdita Plastics, turnover £7 million, has one main manufacturing plant and eight wholesale branch/warehouses across the country. Their products fall into two main categories: there are about 100 'standard' stock items which are stocked at all warehouses; and 'specials' made to customer contract stored in bulk at the appropriate warehouse, and called off by the customer as required.

Perdita has had a computer for a number of years and has accomplished more than most companies. It has some very able computer staff who have successfully introduced computer systems for

Also in 1976


The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN
Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

Mathematical thinking applied to physics...

- deal with multiple ways in
- handle uncertainty
- pick an explicit abstraction
- identify what you don't understand
- and what your options are.

Mrs Diane Perry

1986

I was reading maths.

There was a software crisis.

Some especially important people: Cambridge

Martin Hyland

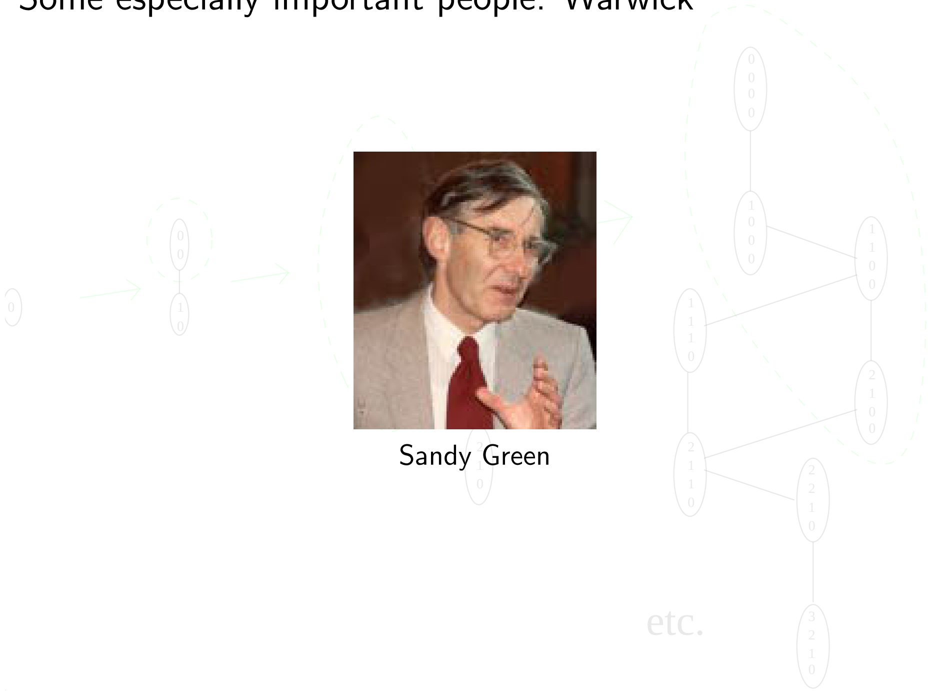
Keith Carne

Julian Bradfield

Graham Weetman

Elsa Gunter

Some especially important people: Warwick

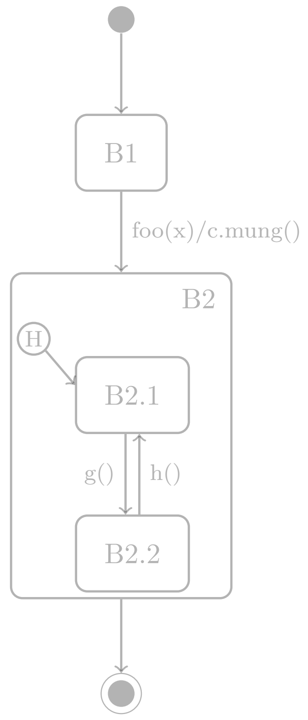


Sandy Green

Some especially important people: BT Glasgow

Tony Hall

1996



I was lecturing software engineering.

There was a software crisis.

1996: Dagstuhl on History of Software Engineering

Peter Shapiro:

In the 1960s, the efficient and timely production and maintenance of reliable and useful software was viewed as a major problem. In the 1990s, it is still considered a major problem. The "software crisis" which was declared three decades ago persists, assuming it makes any sense to speak of a thirty year crisis. Although most would admit to some amelioration of the "crisis," steadily increasing requirements and ambitions have helped sustain it.

At the NATO conferences of the late sixties, the solution to the "crisis" was declared to be "software engineering." This, however, begged a number of questions. What is the nature of software as a technological medium? How does software development compare and contrast with other areas of technological practice. What is engineering? Is it sensible to speak of engineering software?

Answering these questions has been a difficult and tempestuous process which continues to this day.

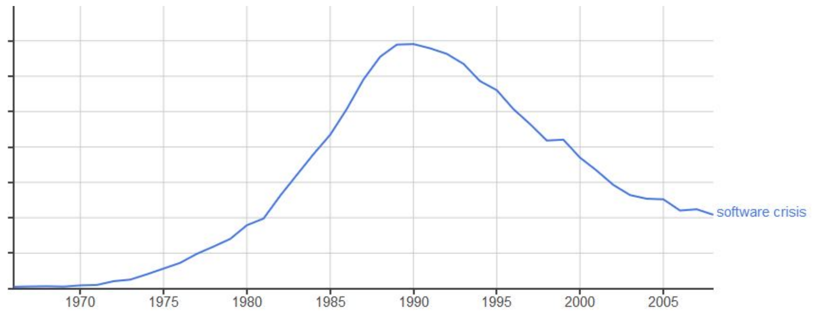
2006

I was... doing lots of things

There wasn't really a software crisis any more.

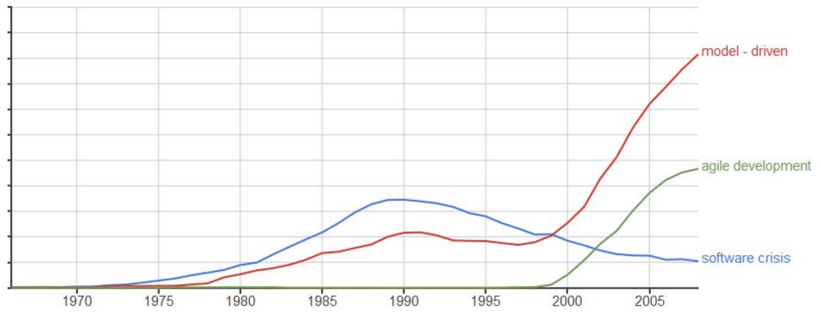
What had happened?

The Rise and Fall of the Software Crisis



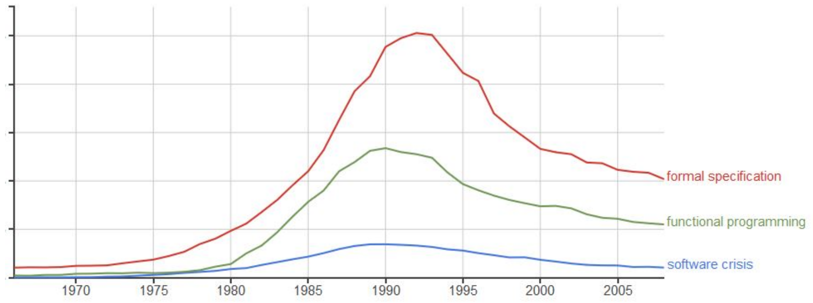
https://books.google.com/ngrams/

Some contributors to the fall



https://books.google.com/ngrams/

Some non-contributors

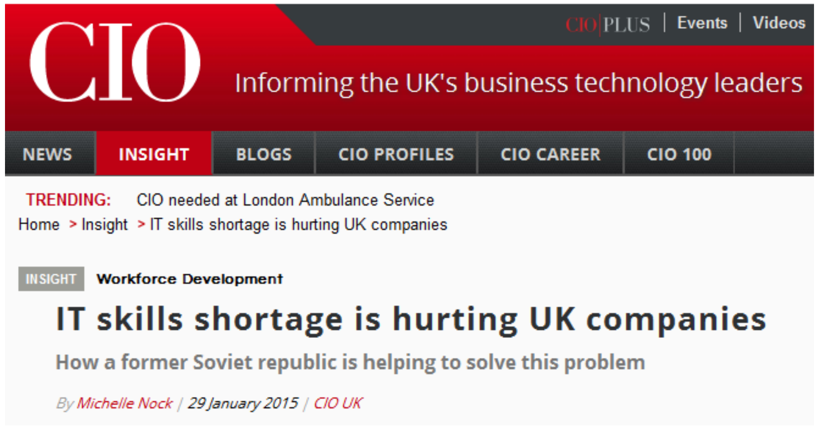


https://books.google.com/ngrams/

2016

Students have no longer heard of “the software crisis”, and yet...

Software crisis, 2016 version


CIO | CIO PLUS | Events | Videos

Informing the UK's business technology leaders

NEWS | INSIGHT | BLOGS | CIO PROFILES | CIO CAREER | CIO 100

TRENDING: CIO needed at London Ambulance Service

Home > Insight > IT skills shortage is hurting UK companies

INSIGHT Workforce Development

IT skills shortage is hurting UK companies

How a former Soviet republic is helping to solve this problem

By Michelle Nock | 29 January 2015 | CIO UK

http://www.cio.co.uk/insight/workforce-development/it-skills-shortage-is-hurting-uk-companies/

CRUNCH NETWORK

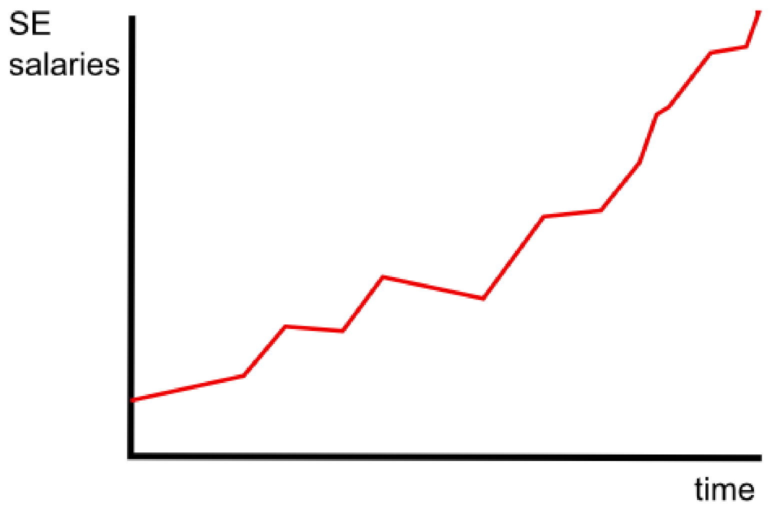
Software Is Eating The Job Market

Posted Jun 9, 2015 by Sean Gallagher (@HiEdStrat)

9,029 SHARES

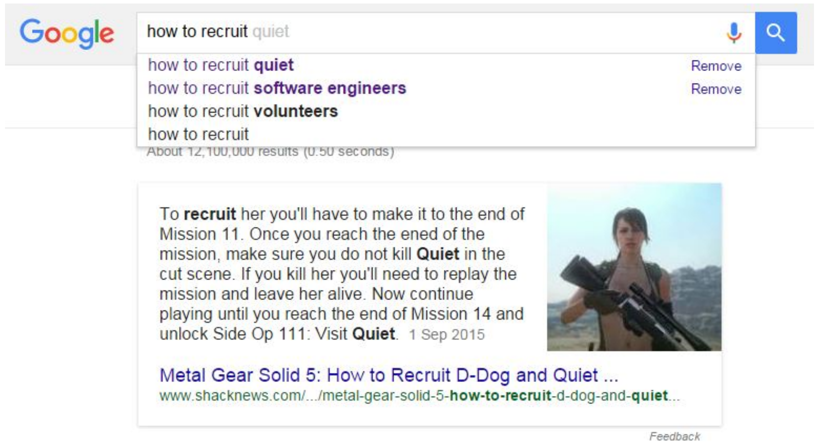
http://techcrunch.com/2015/06/09/software-is-eating-the-job-market/

Seller's market...



(Artist's impression)

...hard for buyers


Google

how to recruit quiet

how to recruit **quiet** Remove

how to recruit **software engineers** Remove

how to recruit **volunteers**

how to recruit

About 12,100,000 results (0.50 seconds)

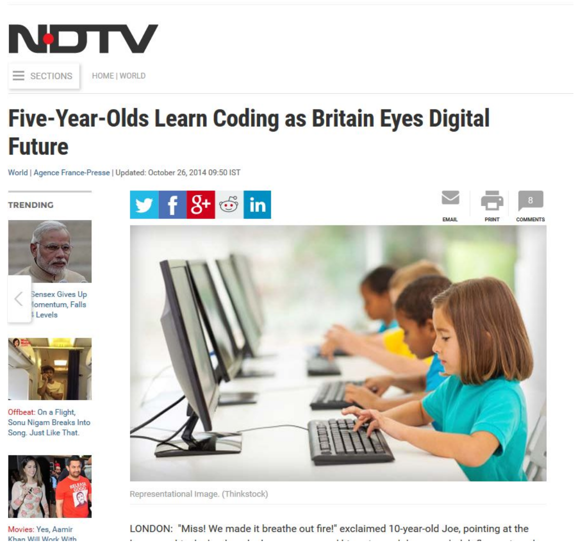
To **recruit** her you'll have to make it to the end of Mission 11. Once you reach the ened of the mission, make sure you do not kill **Quiet** in the cut scene. If you kill her you'll need to replay the mission and leave her alive. Now continue playing until you reach the end of Mission 14 and unlock Side Op 111: Visit **Quiet**. 1 Sep 2015

Metal Gear Solid 5: How to Recruit D-Dog and Quiet ...

www.shacknews.com/.../metal-gear-solid-5-**how-to-recruit**-d-dog-and-**quiet**...

Feedback

So need to increase supply



SECTIONS HOME | WORLD

Five-Year-Olds Learn Coding as Britain Eyes Digital Future

World | Agence France-Presse | Updated: October 26, 2014 09:50 IST

TRENDING

Sensex Gives Up Momentum, Falls Levels

Offbeat: On a Flight, Sonu Nigam Breaks Into Song. Just Like That.

Movies: Yes, Aamir Khan Will Work With

Representational Image. (Thinkstock)

LONDON: "Miss! We made it breathe out fire!" exclaimed 10-year-old Joe, pointing at the

http:
//www.ndtv.com/world-news/five-year-olds-learn-coding-as-britain-eyes-digital-future-684199

Conclusion

Pace Dijkstra, software engineering is more than mathematical engineering;

solving the software crisis required some new ideas.

But now we understand how to build software, and the only remaining problem is that we don't have enough software engineers.

Teach programming in school, and fund lots of SE students, and we'll fix that.

Questions?

I Think You'll Find It's a Bit More Complicated Than That

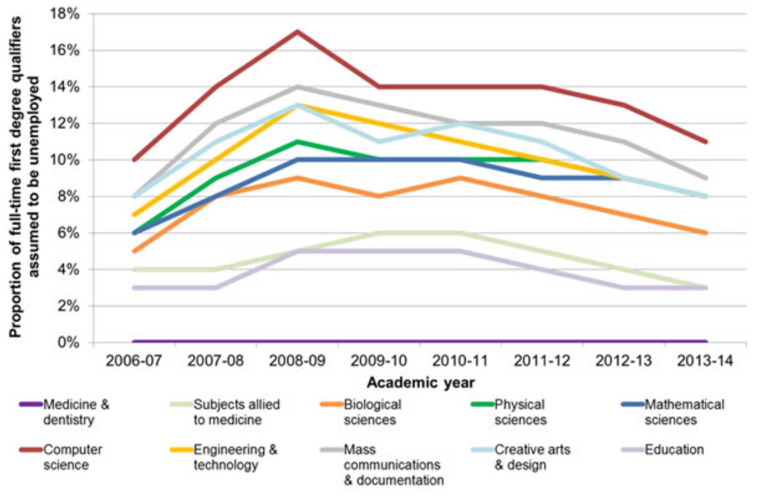
- Ben Goldacre

Below the headlines...

This is backed up by employers such Laterooms.com CTO Stuart Hughes, and Mark Holt, CTO at thetrainline.com. Holt explained recently: "We have a big website and we process an enormous amount of transactions. But, there is a shortage of 'really talented' developers. We have vacancies for as many developers as we can recruit and are receiving many applications for these roles. But while we have been recruiting one or two developers a month for full-time jobs, we are eager to recruit even more - but haven't been able to because of a dearth of talent."

http://www.cio.co.uk/insight/workforce-development/it-skills-shortage-is-hurting-uk-companies/

High unemployment among CS graduates



http://blog.hefce.ac.uk/2015/07/08/unemployment-among-computer-science-graduates-what-does-the-data-say/

High demand for the best CS graduates

While graduates from computer science degrees are more likely to be unemployed, the data also shows that computer science graduates who are in work have among the highest average salaries.

Indeed, what our students can do is amazing.

https://www.thetechpartnership.com/news-events/news/hesa-data-reveals-computer-science-graduate-unemployment-is-double-the-average/

Mismatch? What mismatch?

Perception is that too few software engineers have all the skills necessary.

There are many opinions* on why that is, and what should be done about it.

They boil down to:

- Talent is innate: we just need more potential software engineers to choose from! Get 5yos coding, graduate more students!
- Talent is taught: universities aren't teaching the right things! Teach them Powerpoint/Spring/functional programming/MATLAB/*your favourite silver bullet here*!

Observation: SE success is at least correlated with maths success.

* It's complicated. Shadbolt review...

Maths and...

"Talented" developers must be excellent at

- rigorous computational thinking and
- using current technologies and
- business acumen and
- "soft" skills and
- *your desideratum here*

Ability to abstract, reason, think computationally is essential: and this is mathematics.

If the limiting factor is maths...

then we have a software crisis indeed.

Here at Edinburgh, we are fortunate to be able to recruit very strong students, from across the world, who are very good at maths – and yet, many struggle with mathematical thinking.

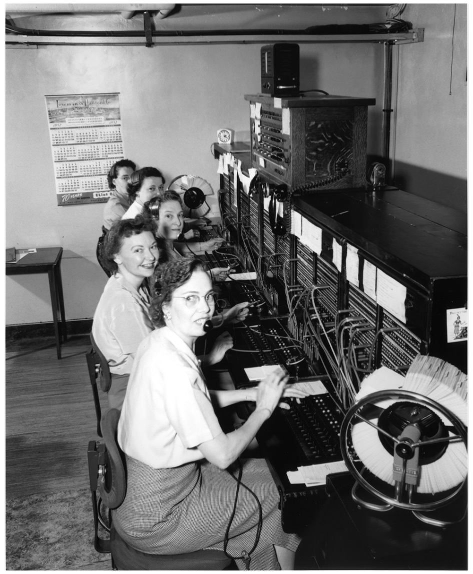
Innate or taught? Hardly matters: we draw on thousands of years' experience teaching maths, we're not going to improve radically now.

So while this is part of what's needed the "talent" recruiters want will never exist in the quantities needed.

We cannot solve this software crisis by growing many more mathematical engineers.

We need to change the game.

1952



Picture: Wikipedia

1902

Picture: Wikipedia. `https://commons.wikimedia.org/wiki/File:Telefonister,_Stockholms_telefonsstation_-_Nordiska_Museet_-_NMA.0037077.jpg`

1886: the telephone crisis begins

William H. Preece, Engineer in Chief of the British General Post Office said that

if the growth of telephone subscribership continued, by the year 2000 every woman in Great Britain would have to be a telephone operator.

The problem had come up fast: ten years earlier he had opined that

the telephone might be all well and good for the United States but that it would never catch on in Great Britain because the country has an adequate supply of messenger boys.

(Both as reported by John Cadogan, Director General, Research Councils of the UK... complex source history!)

Fortunately for me...

Preece was wrong both times.

Photo of a working Strowger exchange at Catford, UK. Photo by Martin Loach

`https://commons.wikimedia.org/wiki/File:Catford_Strowger_Switch.gif`

A hundred years later

"BDUK Broadband (14626513882)" by btphotosbduk - BDUK Broadband. Licensed under CC BY 2.0 via Commons

I did go to work for BT

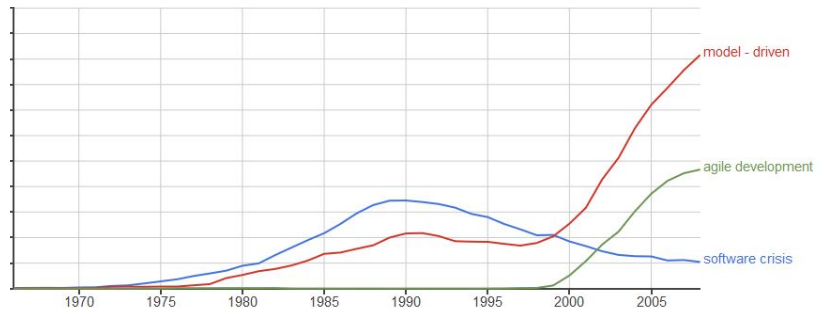
but as a systems and software engineer.

Can we save our five year olds

from having to be programmers?

Or worse, mathematicians?

How did software get so good without x?



`https://books.google.com/ngrams/`

Engineering?

> *Software Engineering was invented as a provocative word without meaning (like "threatened by peace" or "deafening with silence")*

Jochen Ludewig, *Software Engineering – Why it Did Not Work*, 1996

Definition he was criticising:

> *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*

IEEE 610.12

"not a definition, but a goal"

Why is software engineering hard?

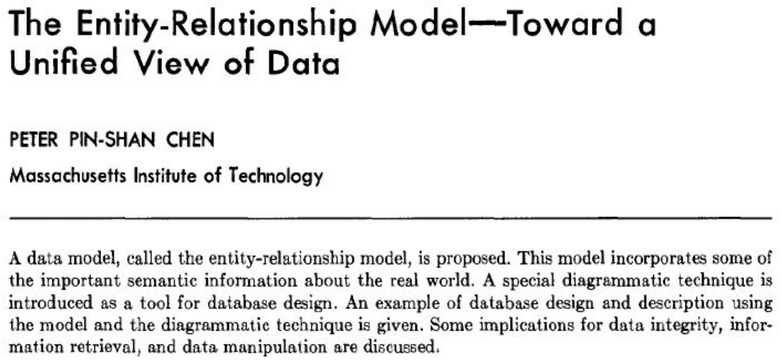
Uncertainty.

- data input – e.g. what value will the function be called on?
- interactions – e.g. in what order will the user press the buttons?
- interleavings – e.g. which thread will win the race?
- change – e.g. what will the customer want next week?

Thinking about large trees of possibilities is really hard.

Origins of MDD

Remember this?



The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN
Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

That paper introduced a modelling language...

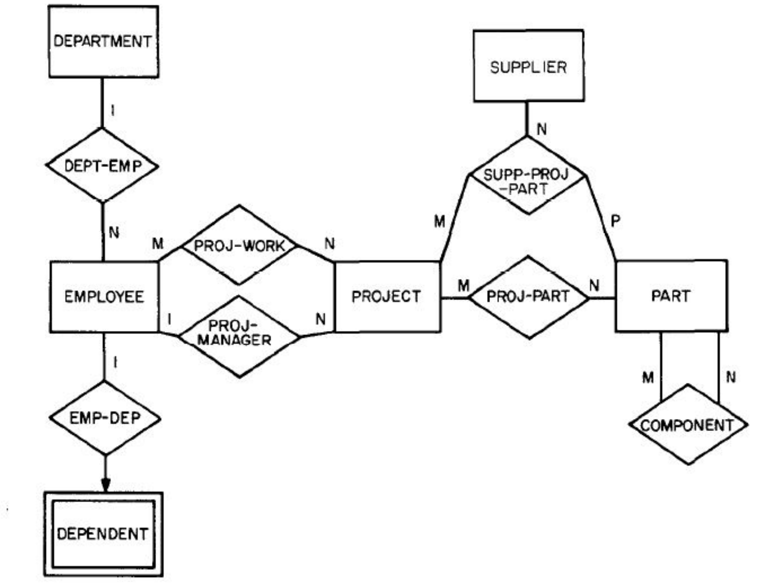


Fig. 11. An entity-relationship diagram for analysis of information in a manufacturing firm

... and note the caption!

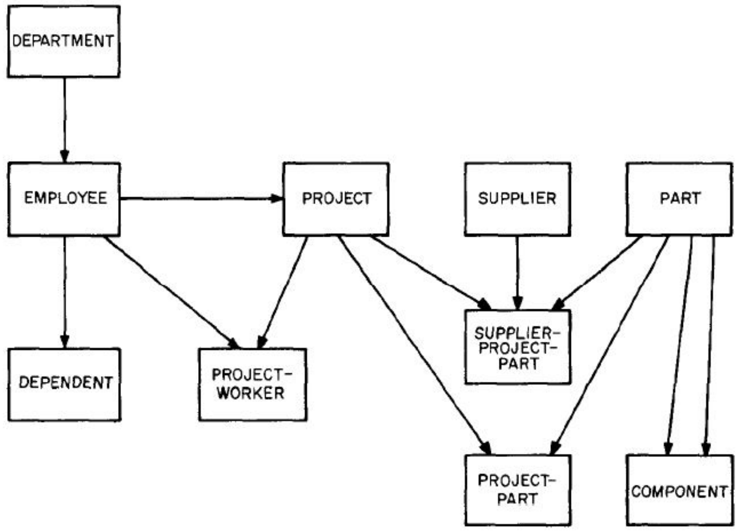


Fig. 20. The data structure diagram derived from the entity-relationship diagram in Fig. 11

Contribution of modelling and OO

Both modelling and object orientation go back to the 1960s, but boomed together in the 1990s.

- abstraction: separation of concerns
- structure the software around relatively stable things.

Model-driven development

Definition

A model is an abstract, usually graphical, representation of some aspect of a system.

Modelling already useful even if models are completely informal.

Definition

MDD is software development in which models are important.

Motivation

Manage information overload, by separating concerns and providing representations suitable for each set of decisions.

Today's MDD: models as formal artefacts

Those who embrace MDD are typically:

- building graphical models in modelling tools;
- in a general purpose modelling language such as UML or SysML;
- or in a domain-specific modelling language, graphical or textual;
- maybe, generating – usually skeleton – code from the model;
- maybe analysing the model;
- maybe generating documentation;
- just possibly doing some round-tripping.

Ideal MDD

Each stakeholder works with a model showing only their concerns.

When their requirements and solutions change, they change their model.

Code is generated from the models.

Verification, validation, documentation, happen at the level of the models.

Yet, MDD enjoys limited success so far.

Cost-benefit problems. Works best with well-understood, predictable development processes, where only one model changes at a time.

Why?

need to keep models/code in sync[1]

code generation in MDE appears, at first glance, to have a positive effect on productivity. But the need to integrate generated code with existing systems may lead to maintenance problems[1]

The majority of MDE users do try to leave generated code alone, but keeping code and models synchronized is clearly an issue[1]

the biggest problem of model-centric approaches is perceived to be keeping the model up to date with the code[2]

A number of informants identified issues of synchronization or consistency as a barrier for wholesale adoption of UML [...] "Need to use it all the way, in order to maintain sync."[3]

Inconsistencies between Software Artifacts. [4]

1. Hutchinson et al.
2. Forward and Lethbridge, *Problems and opportunities for model-centric versus code-centric software development*, MiSE'08
3. Petre, *UML in practice*, ICSE'13
4. Mussbacher et al, *The relevance of model-driven engineering thirty years from now*, MODELS'14

So MDD: problems and promise

Striking similarity between

- trying to introduce computers, in the 1970s, and
- trying to introduce MDD, in the 2000s.

> *Of the comparative studies, results vary between a 35% gain to a 27% loss. Two of the studies showed no net impact. [Yet] some of the productivity gains described are far in [excess] of those discussed earlier: 2x, 5x and even 8x productivity improvements.*

Empirical Assessment of MDE in Industry Hutchinson et al. ICSE'11

Agile development

Definition

Agile development is a philosophy of software development that aims to embrace change.

`http://www.agilemanifesto.org/principles.html`

structure the development process round the customer's priorities

Motivation

Manage information poverty, by not waiting for full information.

Agile development has problems too

- Architecture
- Maintenance
- Need for exceptional programmers.

See e.g.

- `http://www.cio.com/article/2385322/agile-development/why-agile-isn-t-working--bringing-common-sense-to-agile-principles.html`
- `http://www.infoworld.com/article/2624279/agile-development/agile-programming-10-years-on--did-it-deliver-.html`

Suppose we could do both at once?

From MDD: separation of concerns

From agile: keeping pace with the real world

> *The art of progress is to preserve order amid change and to preserve change amid order.*
>
> Alfred North Whitehead

Be careful what we wish for

Currently MDD requires you to be a mathematician, agile requires you to be an ace programmer. Neither leaves space for you to be expert at something else.

Current attempts to combine agility with MDD threaten to require you to be both mathematician and ace programmer.

That's not necessarily an improvement...

Vision

Disclaimer: I'm talking about 50 years in the future here, and 50 years ago, people didn't foresee today!

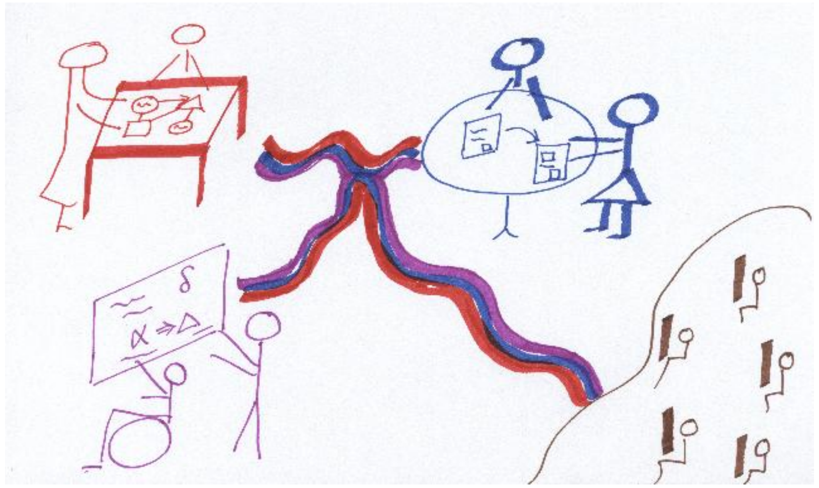
Hard parts of SE are requirements, and maintenance...

... and I don't work on either. Mathematics has no attack there. But...

...Suppose software experts didn't understand customer requirements, but the kinds of things that these customers have requirements for. And suppose customers did their own maintenance.

That should be the ultimate vision of agile MDD.

Software development in 2066?



Bidirectional transformation languages

Problem: it's really, really hard to write the wires.

Solutions?

- better languages and tools to support that. Let's dig down...

Unidirectional transformations

generate one model (including code) from another:

$$f : M \to N$$

We have fairly decent languages to express unidirectional transformations.

Problem: when you change the target model, you're stuffed.

Bidirectional transformations

A bidirectional transformation (bx) needs to be able to:

- check consistency

$$R \subseteq M \times N$$

- restore consistency:

$$\overrightarrow{R} : M \times N \to N$$

and dually

$$\overleftarrow{R} : M \times N \to M$$

with sensible behaviour, e.g. being

- correct: consistency restorers really do restore consistency;
- hippocratic: if nothing needs to be done, do nothing.

Two intertwined problems

1. what should a bx do?
2. how can we engineer confidence that it does that?

Bidirectional transformation languages

Problem: lots of duplication

Solutions?

- bx languages

We have languages that can express bx, but no good ones yet.

Bidirectional transformation languages

Problem: leave the nice world as soon as you want interaction, non-determinacy, etc.

Solutions?

- bx languages with effects

Abou-Saleh, Cheney, Gibbons, McKinna, S. *Notions of Bidirectional Computation and Entangled State Monads* MPC'15

Why is software engineering hard?

Uncertainty.

- data input – e.g. what value will the function be called on?
- interactions – e.g. in what order will the user press the buttons?
- interleavings – e.g. which thread will win the race?
- change – e.g. what will the customer want next week?

Thinking about large trees of possibilities is really hard.

Verification games: handling an uncertain future

To show some property *always* holds:

Verifier who wants to show it does

plays

Refuter who wants to show it doesn't

Refuter challenges the system in whatever ways we want to allow: if Verifier can always meet those challenges, the system is correct.

This idea has a long history. Where I came in:

S., Stirling *Practical Model-Checking Using Games*, TACAS'98

S., *Abstract Games for Infinite State Processes*, CONCUR'98

Exploration games: handling uncertain requirements

$$\begin{array}{rcl} \text{challenges} & \longleftrightarrow & \text{requirements} \\ \text{responses} & \longleftrightarrow & \text{design} \end{array}$$

Design the game itself... refine requirements and design together, stop when both are precise enough and consistent.

Tenzer, S. *GUIDE: Games with UML for Interactive Design Exploration*, JKBS 20:7, 2007

MDD games: handling uncertain future requirements?

Can we combine those ideas? Then

- to specify a bx is to design a game (DSMLs; consistency defn; properties of consistency restoration)
- to write a bx is to design a winning strategy for that game (how will consistency be restored?)
- to apply a bx is to play the game according to that winning strategy.

Early steps:

S. *A simple game-theoretic approach to checkonly QVT Relations*,SoSyM 12:1 2013

Bradfield, S. *Recursive Checkonly QVT-R Transformations with General when and where Clauses via the Modal Mu Calculus*, FASE'12

Bradfield, S. *Enforcing QVT-R with mu-Calculus and Games*, FASE'13

Bidirectional transformation languages

Problem: correctness and hippocraticness aren't enough to rule out weird behaviour

Solutions?

- bx languages with Principle of Least Surprise
  – based on providing reasonable behaviour in senses long understood by mathematicians, e.g. Hölder continuity
- and notions of atlas, and tools that go BEEP?

Cheney, Gibbons, McKinna, S. *Towards a Principle of Least Surprise for Bidirectional Transformations* Bx'15

Bidirectional transformation languages

Problem: even if we develop a bx language in which all bx are reasonable, how can we tell that they are correct?

Solutions?

- testing techniques for bx
- verification of bx, especially lightweight verification – types
- composition and reuse

Bidirectional transformation languages

Problem: models might change *while the bx is running*. Especially if models are large and the bx is slow...

Solutions?

- parallel semantics for existing bx languages?
- dedicated parallel bx languages?
- what guarantees can we make?

DWIM (do what I mean) interfaces

Problem: even if we give domain experts a beautiful model with only what they're interested in... if we treat that as a formal object, we are, essentially, asking them to do programming.

Solutions?

- perhaps it isn't really a problem at all? Perhaps if it's beautiful enough, it doesn't feel like programming?
- perhaps our system can be clever enough to work out, or elicit, what the domain experts actually want?

Alta Vista 1996 vs Google today

DWISM (do what I should mean)?

Problem: domain experts, or people who play them, might be wrong or even malicious.

Solutions?

- perhaps the benevolent DWIM system will simply raise a puzzled eyebrow?
- more familiarly, verification?

What has mathematics to do with software engineering?

Perhaps:

Software engineering is a branch of applied mathematics – but the devil's in the application.

Perhaps:

Mathematics is why it's hard to find enough good software engineers.

Perhaps:

Mathematics is what will allow us to widen participation in the building of software.

Special thanks

- to my colleagues on project *A Theory of Least Change for Bidirectional Transformations*:

Faris Abou-Saleh, James Cheney, Jeremy Gibbons, James McKinna

- to my friends and family

- and to everyone who continues to teach me.

Questions?

Mathematics is...

understanding the structures of things and how they change

Software engineering is...

understanding the structures of things and how they change

Maxims

If at first you don't succeed, find the mistake.

How hard can it be?

What does it have to do with the price of fish?

Never attribute to malice what can be explained by mere incompetence.