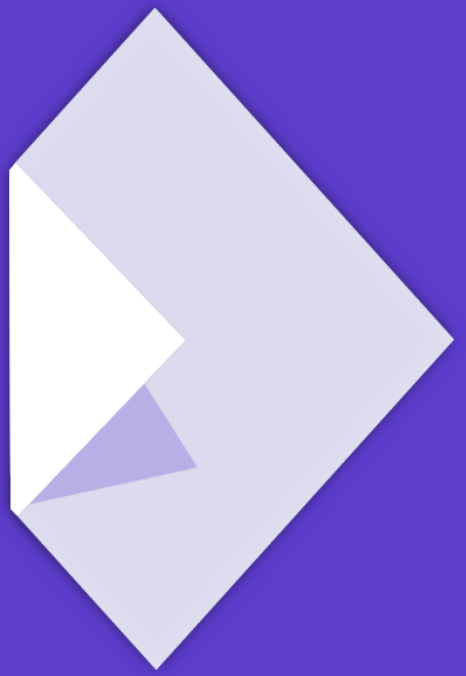
Thoughts on skills needed for a career in Software Engineering

NB - not Computer Science



Collabora Productivity

By Michael Meeks

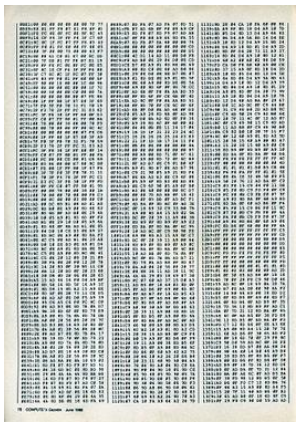
General Manager, Collabora Productivity

@mmeeks @CollaboraOffice

"Stand at the crossroads and look; ask for the ancient paths, ask where the good way is, and walk in it, and you will find rest for your souls..." - Jeremiah 6:16

Why you might want to listen.

5 years – hobbyist programmer – the type-it-in computer game:



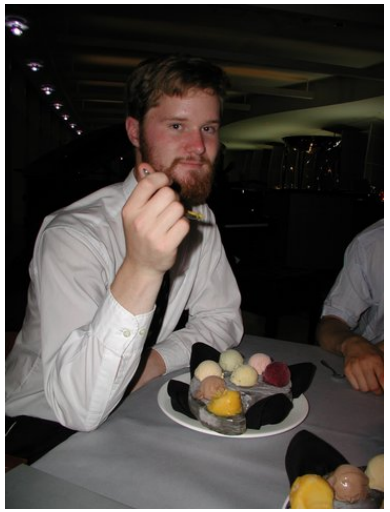
25 years worth of mistakes on someone else's $

- Hardware design board, enclosure, FPGA
- RTOS drivers → FLOSS: Kernel → User-space → GUI

As an engineer

- travelled the world on someone else' dime ($, £ etc.)
- given hundred+ conference talks, collaborated with hundreds of sharper engineers than myself.
- interviewed hundreds of people, hired a hundred or so.
- tech-edited books, magazine columns, standards committees, software & consultancy sales …
- **made a epic mess of lots of things.**

Software Engineering is just typing ?



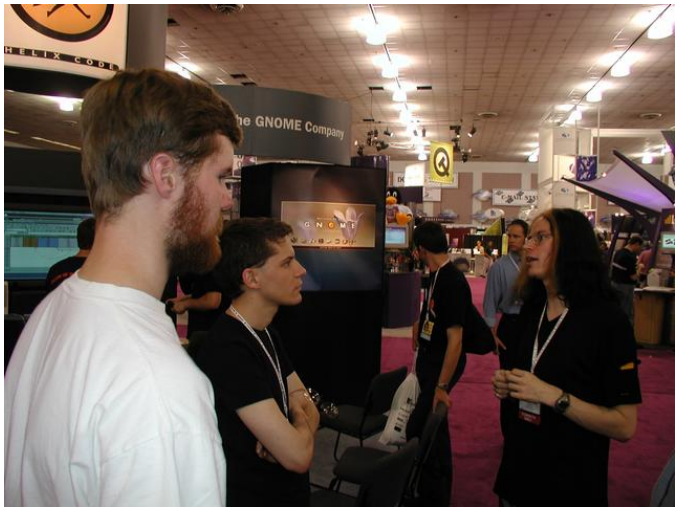
Seoul South Korea

Norway

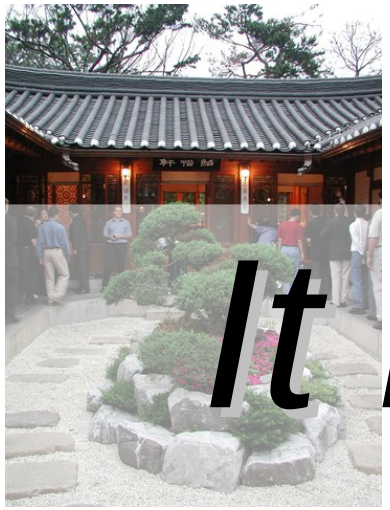
Plus:
Dublin,
Hamburg
Ottawa
...

Paris:

San Francisco



Just my graduation year: ~2000



Seoul South Korea

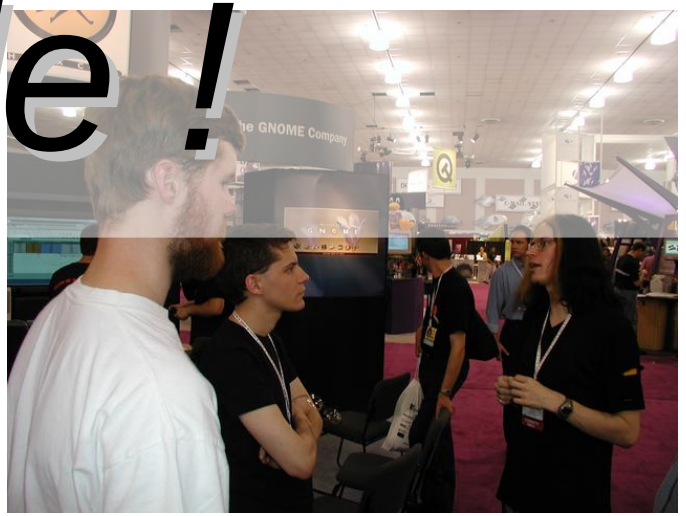
Norway

It is all about the people !

Paris:

Sa
Ea

collabora



Plus:
Dublin,
Hamburg
Ottawa
...

Skill #1 – people skills – they matter.

Engineering doesn’t happen by itself

[people] think technology **just automatically** gets better every year, but it actually doesn't.

It only gets better if **smart people** work like crazy to make it better. That's how any technology actually gets better.

And by itself technology … if people don't work on it, it actually will decline.

How to build the future Elon Musk (YCombinator)

Skill #2 – self motivation.

Self motivation – some thoughts:

Learning at Hills Road

- Good: no-one is going to do it for you.

Learn Autonomy

- Be an ideal employee:
  - “Go! Do good things!
    Then tell me about them!“
- do this and you will thrive.

Show up & work hard

- Do that and you’re better than avg. 50-70% of the staff.
  - Weekend job ? …

Someone else is not going to do it for you

- In our industry, you hit the boundary of ignorance fast …
- People create complexity: lots of it.
- Strive for excellence – vote for a wealthier society.

Computer Science too easy ?

- **Compound your Interest**
- Life is like a race: run 2% faster than the next person – and over a year ...

Create or consume ...

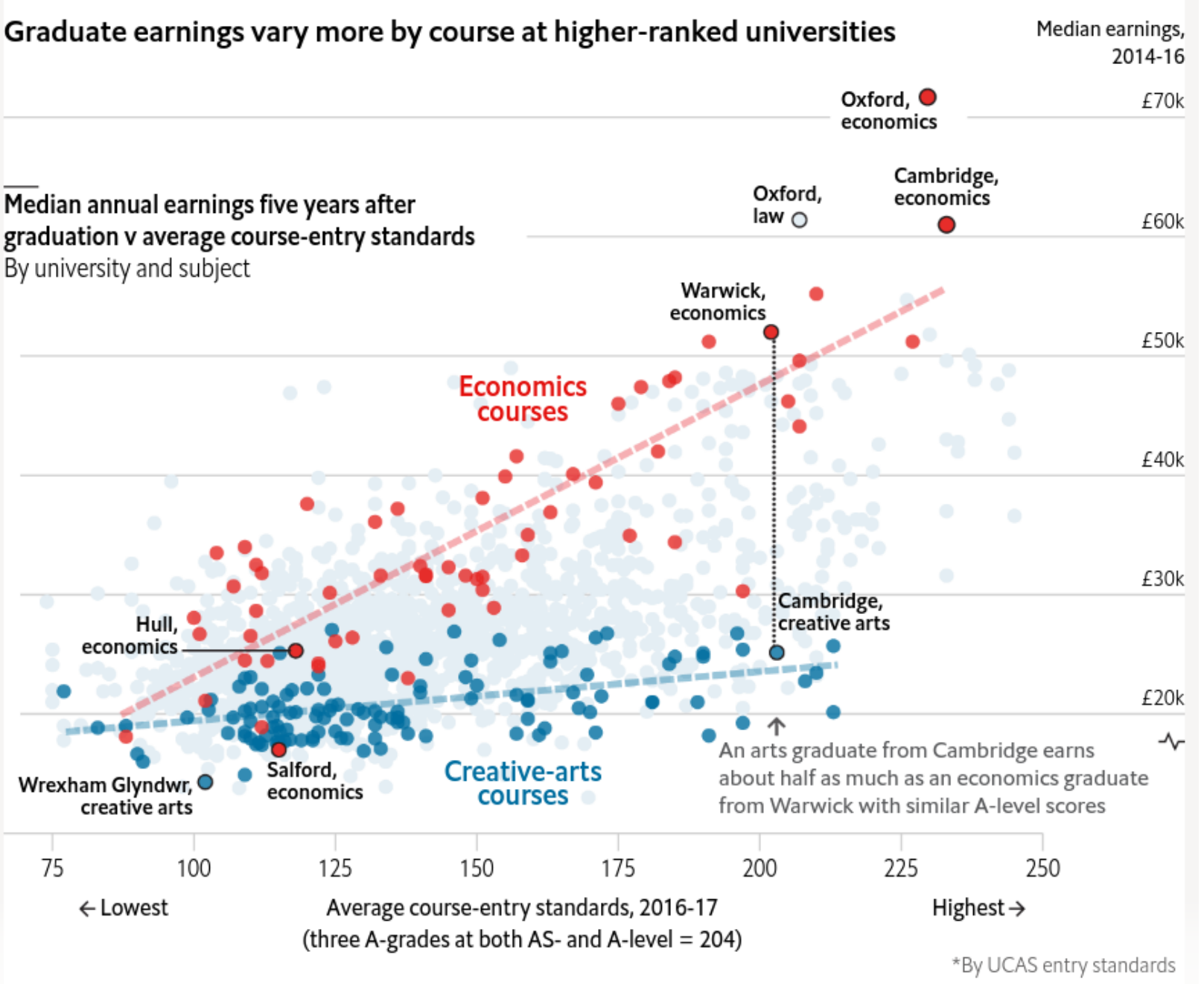
- Stop playing computer games
- Start writing them; obviously.

The Economist

Software Engineering – in the middle there somewhere:

It is worth working harder now to get a better degree.

cf.
Source

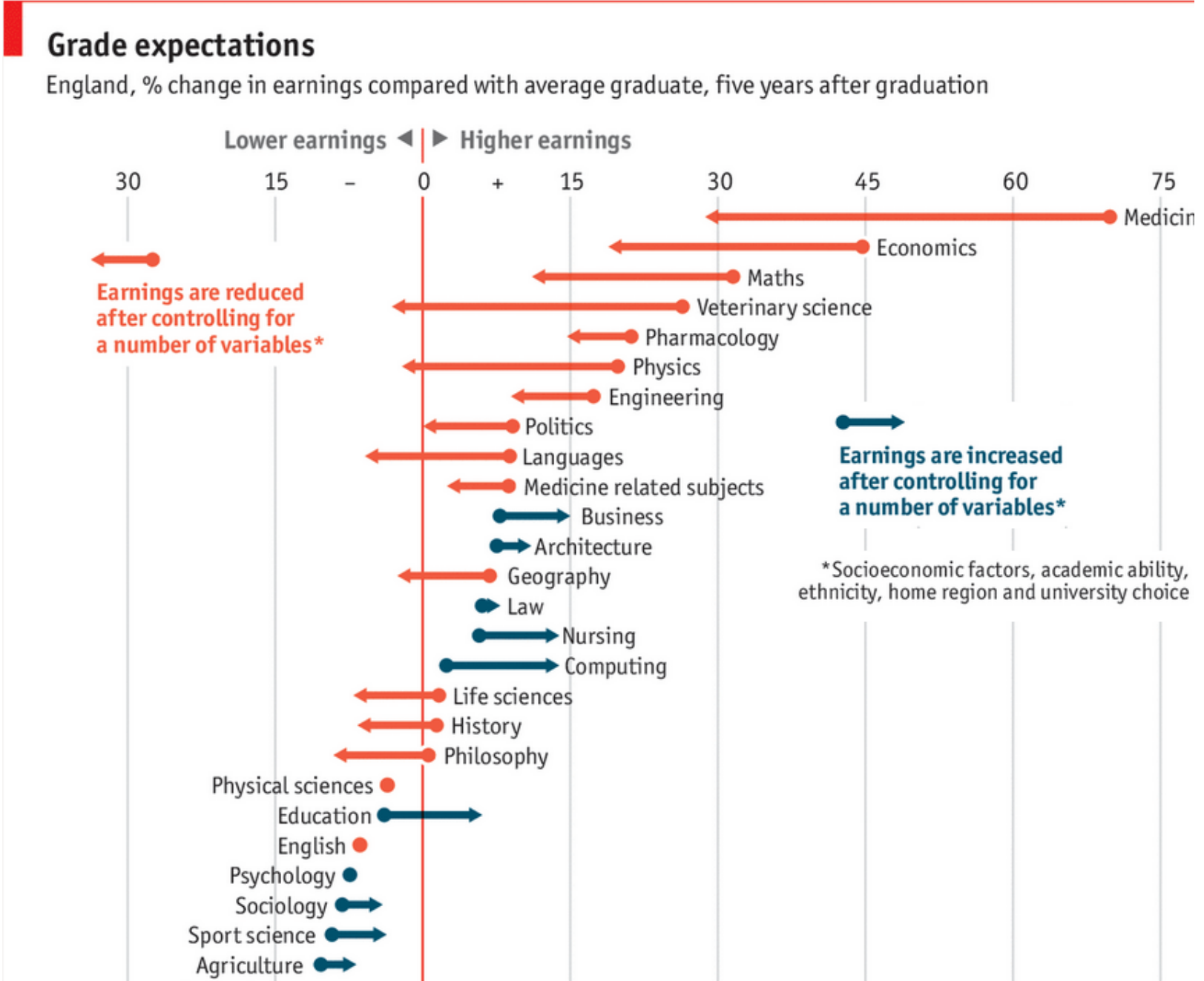


The Economist

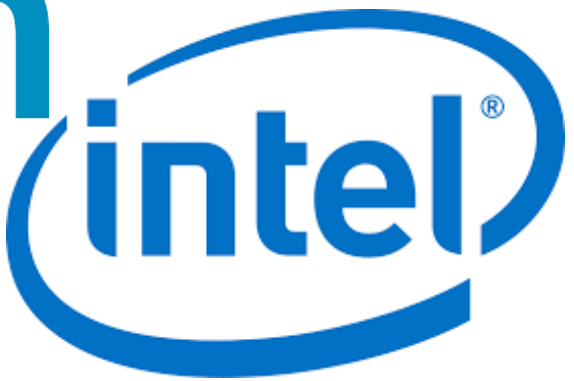
Computing: earnings adjusted upwards when controlling for socioeconomic / ethnicity / geography:

ie. earnings can't be explained just by being bright & white.

Source



Consider the alternative …



Imagine tomorrow:

- Everyone at these companies decides to become ski-instructors or insurance salespeople & quits.
  - in ~30 days – there are no chip companies.
- We all have to stop bit-coin mining & do something useful

Don't treat companies as monoliths

- They are really a conglomorration of warring tribes
- If: Aleksandr Solzhenitsyn is right: *"The line separating good and evil passes not through states, nor between classes, nor between political parties either -- but right through every human heart"*
  - Then viewing a company as monolithically evil is an even deeper folly.

Work hard – but smart: “Be lazy”

How to stop chunks of ice forming on the nose cone

- They tend to fall into the extremely expensive jet engine with incredibly tight tolerances

Easy – we have a furnace just back here

- Complex system of tubes, pipes, pumps, fluids, inside a giant spinning cone:
  - Job done! Modulo maintenance

Or …

- A special coating on the nozzle
  - Ice falls off in small pieces …

Credit: Dysanovic, CC

Skill #3 – re-write to simplify throw away your code !

Less code is better code:

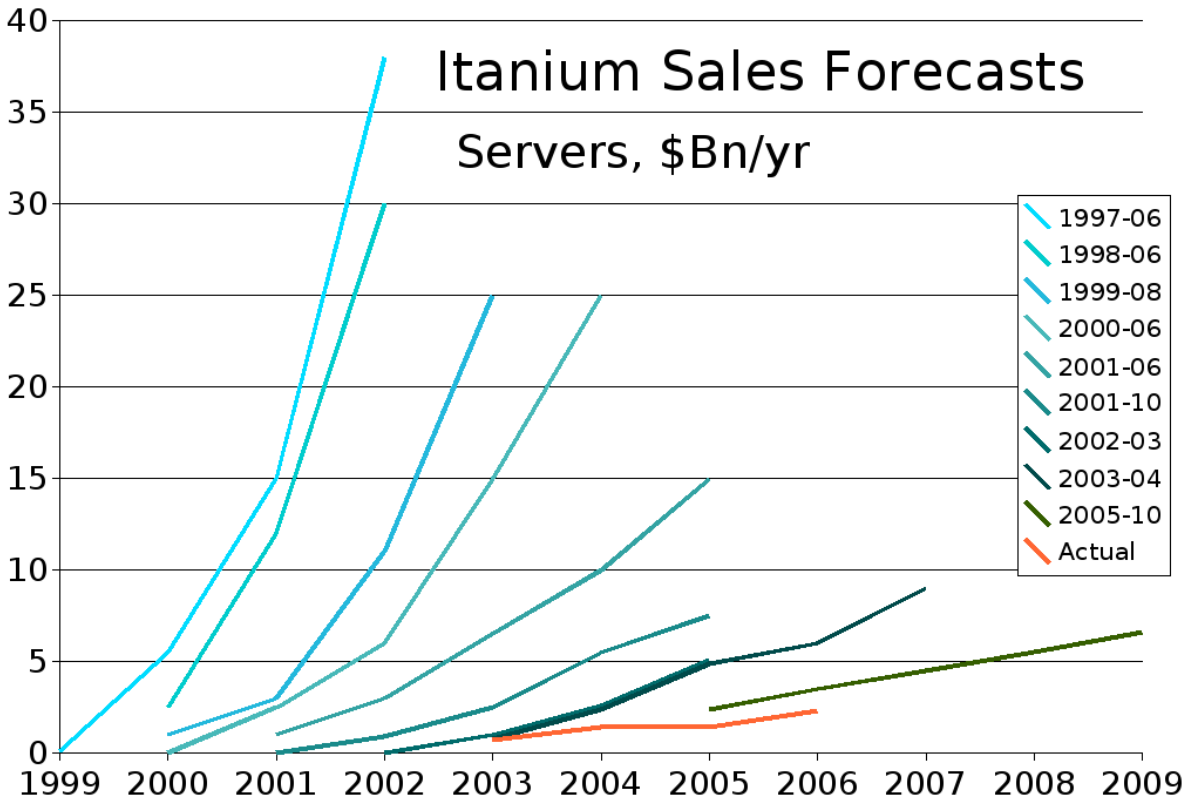
*"Measuring programming progress by lines of code is like measuring aircraft building progress by **weight**."*

Bill Gates

But – it is hard to discard code: revision control is your friend … it's still there (somewhere)

Example: AMD64 vs. Itanic

Multi-billion dollar investment in a new 64bit processor architecture: revolutionary EPIC, VLIW, compiler-exposed-parallelsim: re-write the pile of TCL that is your Intel CPU.



Credit: Arch dude CC BY-SA 3.0

AMD64 (or “x86-64”) - simplified

What are the problems with x86 ?

- Not 64bit
- Almost no registers (~8)

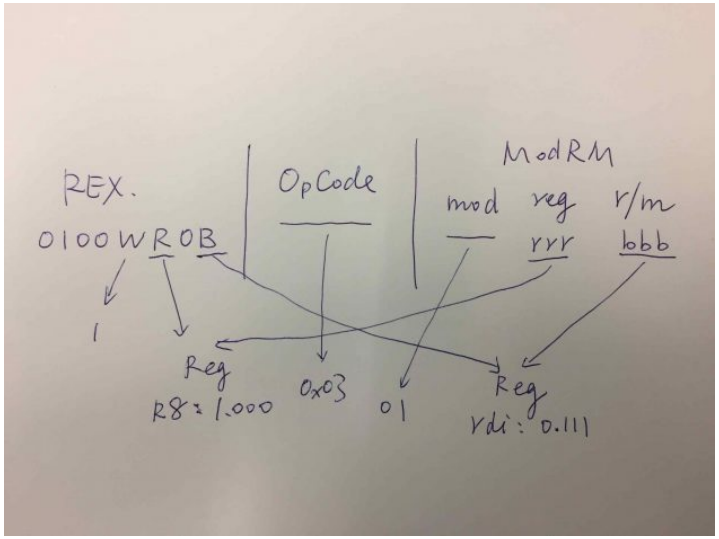
How can we most easily hack 64bit into the Instruction Set Architecture ?

- Some instructions use three registers: huh … **CISC** – right ?

```
/* Move the 4 bytes of data at address ESI+4*EBX into EDX. */
mov (%esi,%ebx,4), %edx
```

To double the # of registers we need 3 more bits … how do we do that ?

AMD64 (or “x86-64”) - simplified



SysTutorials

Idea ?

- Wait – we have a lots of ‘old’ one-byte op-codes: inc (0x40+r), dec (0x48+r), etc.
- If we steal these …
  - We can use them as a ‘prefix’ that makes the following instruction a ‘64bit’ instruction and 8 gives us 3 new bits we can use to address our registers.
- Yes the compiler / assembler will have to work hard
  - But not as hard as an Itanic / EPIC / VLIW nightmare.

The quick hack → beautiful – total market dominance.

Skill #4 – Technique

Something anyone can
learn – but no-one wants to.

Abandon intuition & trying – it only goes so far

Computers behave deterministically

- They are comprehensible … but.
- Very large problems are very tricky …

You're doing a project soon ?

- First read: *"Solving arbitrary problems from a standing start"*
  - Or *"Real World Engineering"*

Best here:

https://people.gnome.org/~michael/data/2016-04-29-solving-problems.pdf

Why you need to care ? #2

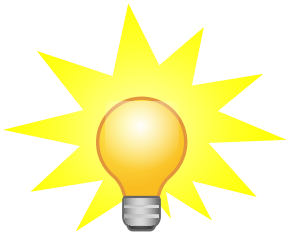
Technique tends to be either

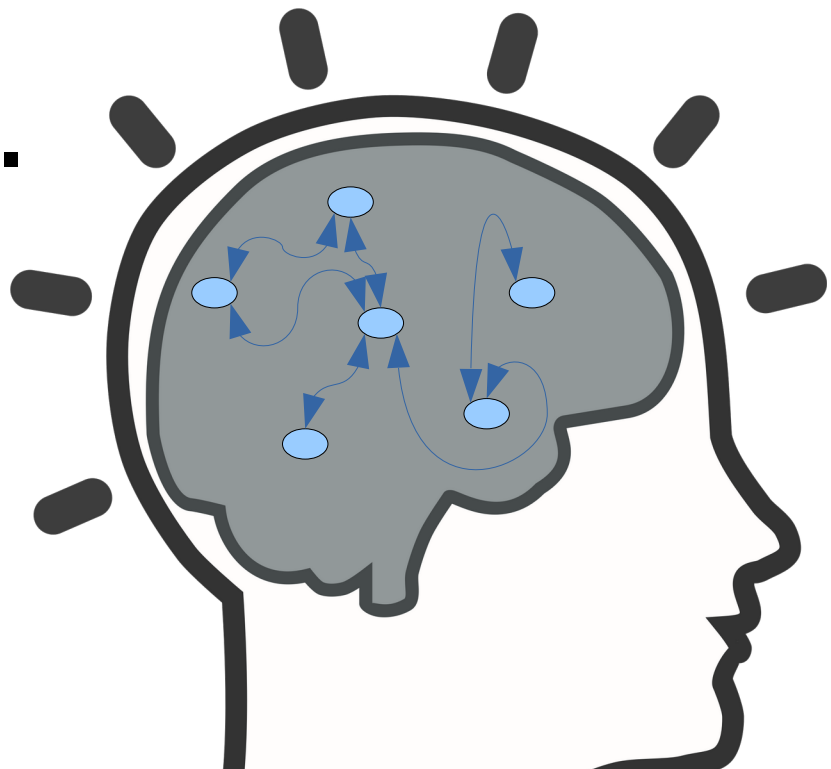
- ‘intuition’ or understanding the
- Whole problem:

fitting it all in your mind.

“If I have two kinds of fish, and they are in the ratio 3:2 big to small, and there are 4 small fish, how many do I have ?”

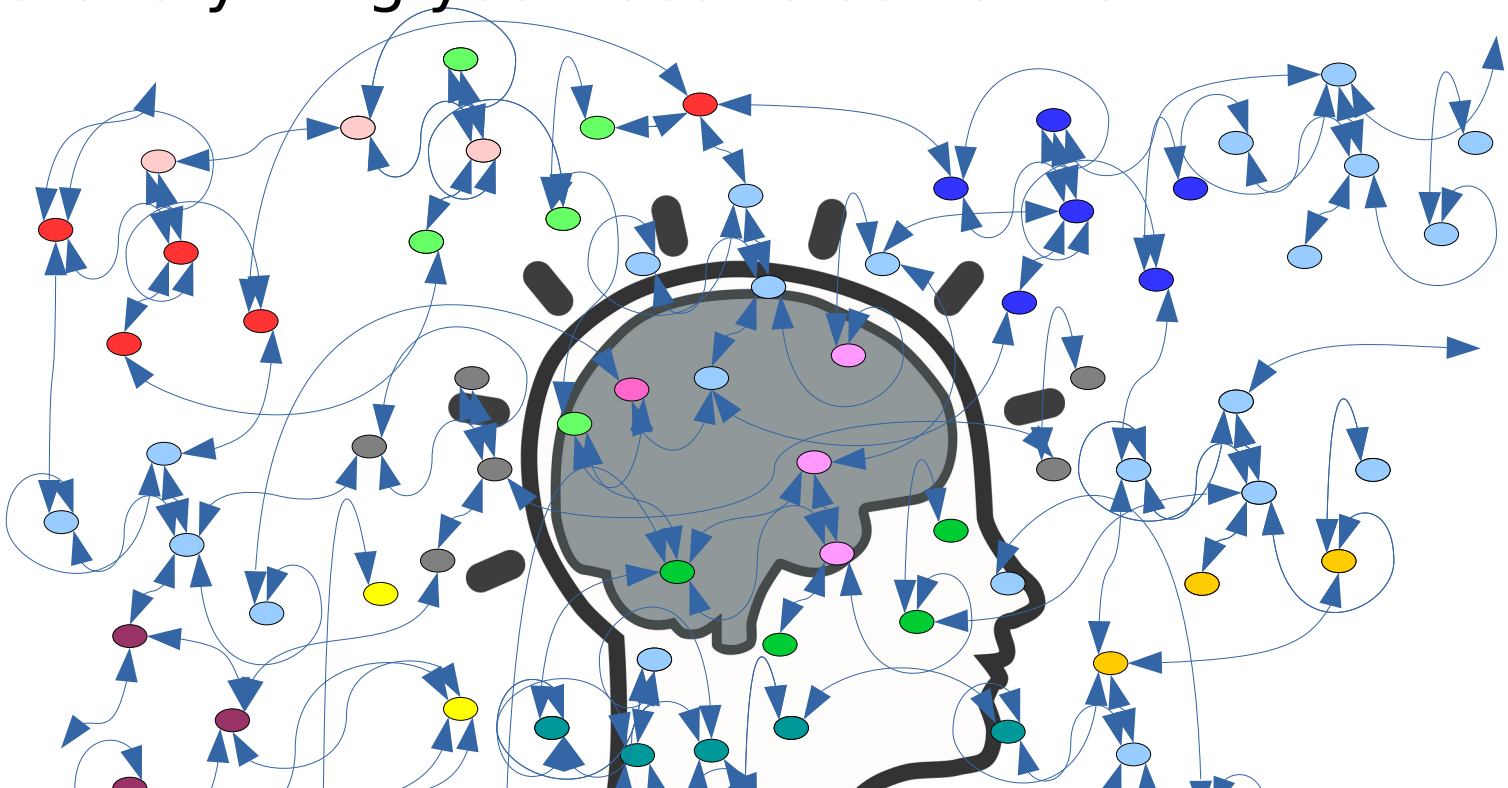
“It’s obvious” → solution ...





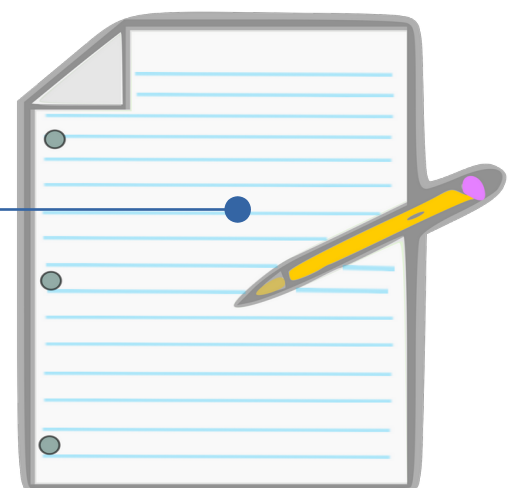
Not all problems are simple:

- Smart people often delay learning scalable technique – some never learn & plateau
- You can’t hold everything you need to solve the problem in your head.
- Need a scalable technique.



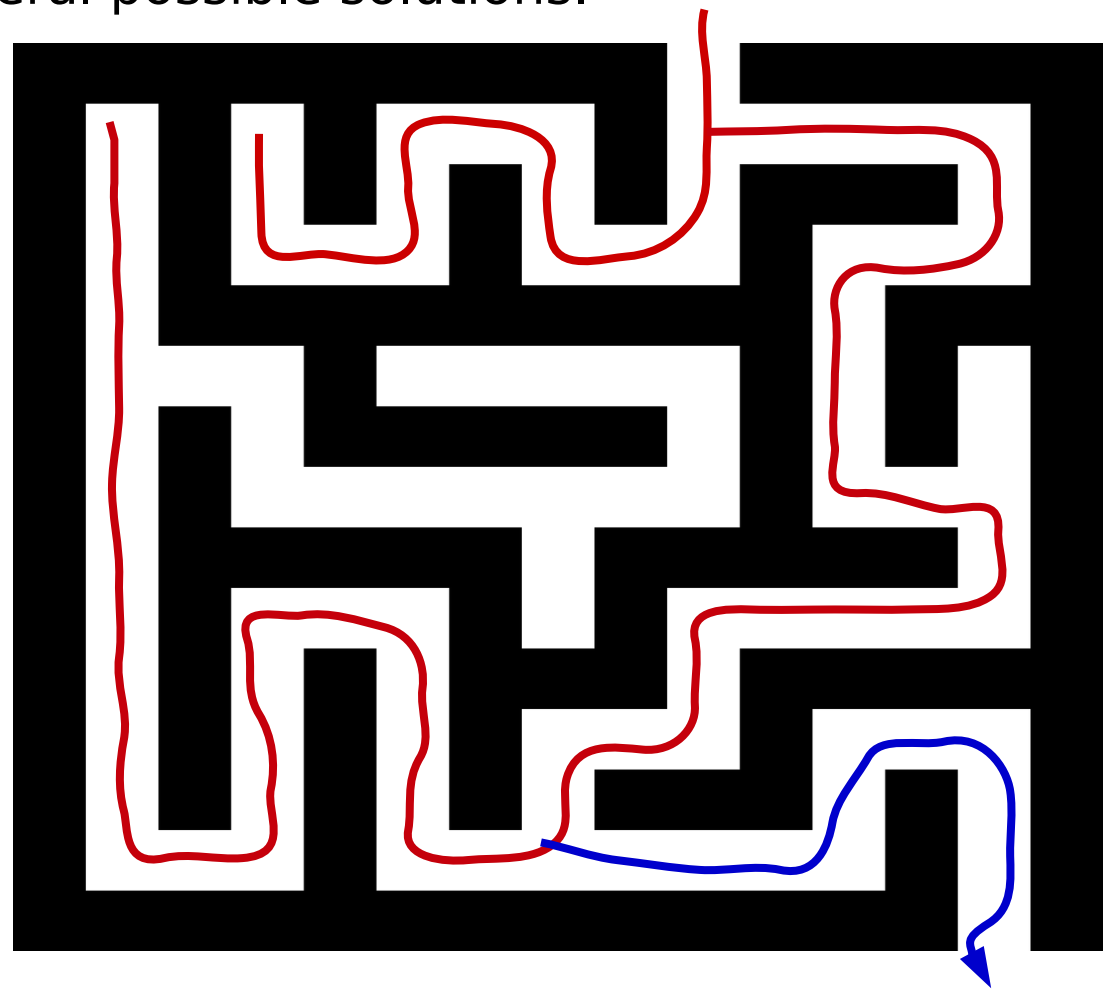
Reductionism is your friend

- Give up on the initial “intuition” technique.
- *“A group of footballers come from Ankara, Amasya and Aydin. Seventeen more come from Aydin than from Amasya, twice as many come from Ankara than from Amasya, there are 31 footballers from Aydin – how many are there in total ?”*
  - A trivial problem – perhaps you can do it in your head ? Are you sure your answer is correct ?
  - How about if I add another couple of towns & numbers ?
- The magic tool:
  - Break the problem into small steps
  - write them down
  - check them.



Why write things down ?

- It is normal to evaluate several possible solutions.
- Before discarding all but one or two and proceeding with those ...
- Document your assumptions & choices so you can back-track without getting lost.
- Average time to fix a on-trivial bug in LibreOffice: 1 week.



Skill #4 – Technique

Can only be learned by tackling problems that you cannot solve without it.

The pre-eminent engineering skill.

Skill #5 – Extreme Focus & Sustained Attention

Monomanical Focus … good ?

Ignore the person pouring liquid nitrogen; I'm hacking!

Neurotic Perfectionism ?

- A super-power you need to learn to control
- Not got it – practice!

Perspective also vital:

- Breadth or Depth ? - a bit of both.
- Rat-holes / Bike-shedding

Quit typing – and put it on the mental after-burner …

Skill #6 – Compromise & Collaboration

With all that wonderful focus …

Communication skills

- Compromise
- *“I can do it better”* – but can you really on your own ?
  - Will you ultimately enjoy it as much ?

Project / pairing

- Find someone very different from you …
  - Not so you can do it all and they can watch
- So you can learn to collaborate.

How to treat a new developer's code ....

Remember this patch / feature is their baby ...

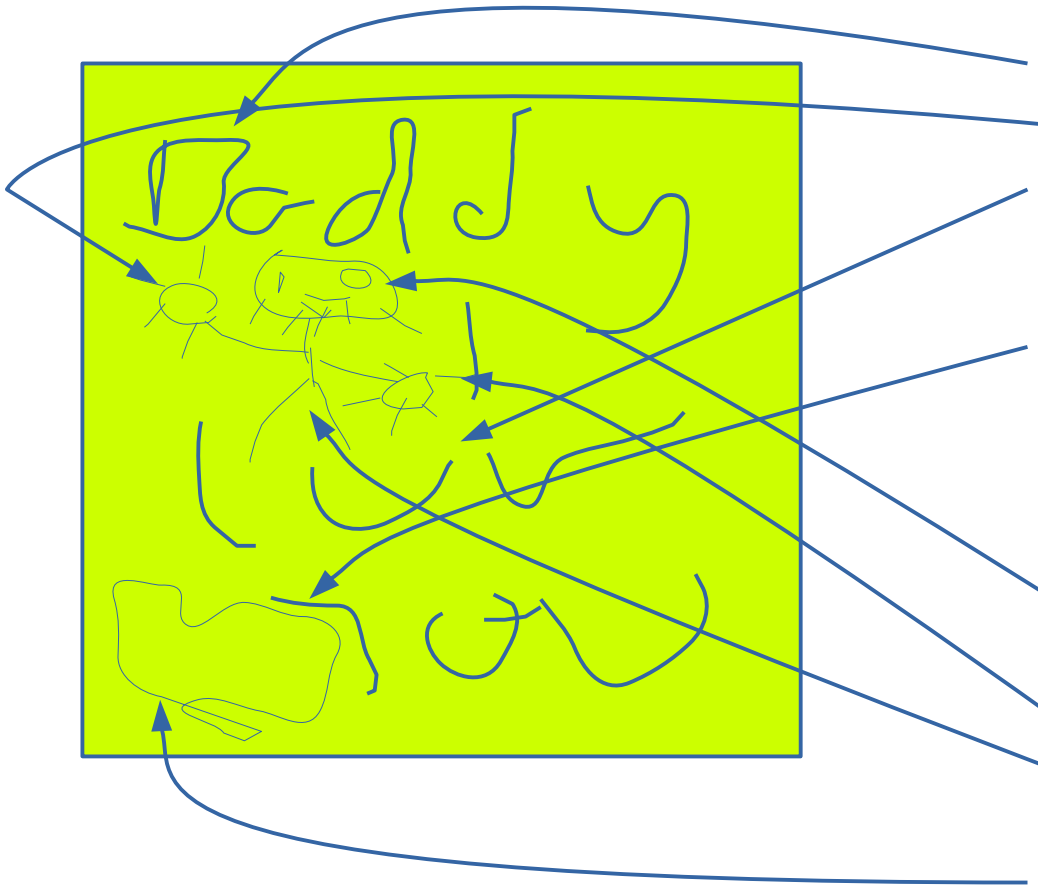
Priorities / Patterns

- Nurturing and encouragement
- Immediate gratification / merging
- Helping them out with the next step
- Zero review / round trip latency
- Praise/Criticism/Praise sandwich

Anti-patterns

- Extensive public criticism of babies, better-but-not-perfect
- Asking open-ended / vague / hard-for-you-to-answer questions:
  “does it defurbulate VCL's crudgenickle sklep ?”
- Non-concrete action mails of vagueness

Example one: how not to do it ...



'D' in 'Daddy' looks like a 'B'

People have five fingers

Love is spelled 'Love' not 'Luv'

Y in 'you' looks like a squiggle

Page layout is terrible.

Wait – is that me !?

- My eyes are the same size
- My hands are not potatoes
- My body is much thicker !

Hearts are not this shape

With thanks to Adrian Plass

Perhaps a more winsome approach ...

Wow !

- You love me :-)
  - I love you too ...

Thank you so much for making that for me

- I'd really like to meet up and practice some of the hard letters sometime

I particularly like the smile

I've pushed it with a few minor corrections to the master branch

Thanks again & looking forward to your next piece.

Working with others:

Riches of Social interaction

- If an amazing technical breakthrough is made (in a forest) and there is no-one but you to appreciate it – did it happen ?

Linus Torvalds I recall well in early ~2000's

- got frustrated with giving interviews, utterly bored:
- Kept overhearing in speakers lounges things like this:
  - ***"Linux is worth $<N> billion – why did you make it public, when you could have kept that all for yourself ?"***
- Hard to see how you can miss the point any further.

Skill #6 – Compromise & Collaboration

But I can do it all myself! - beyond ‘Hello World’

Software Engineering has many ~orthogonal axis (at least):

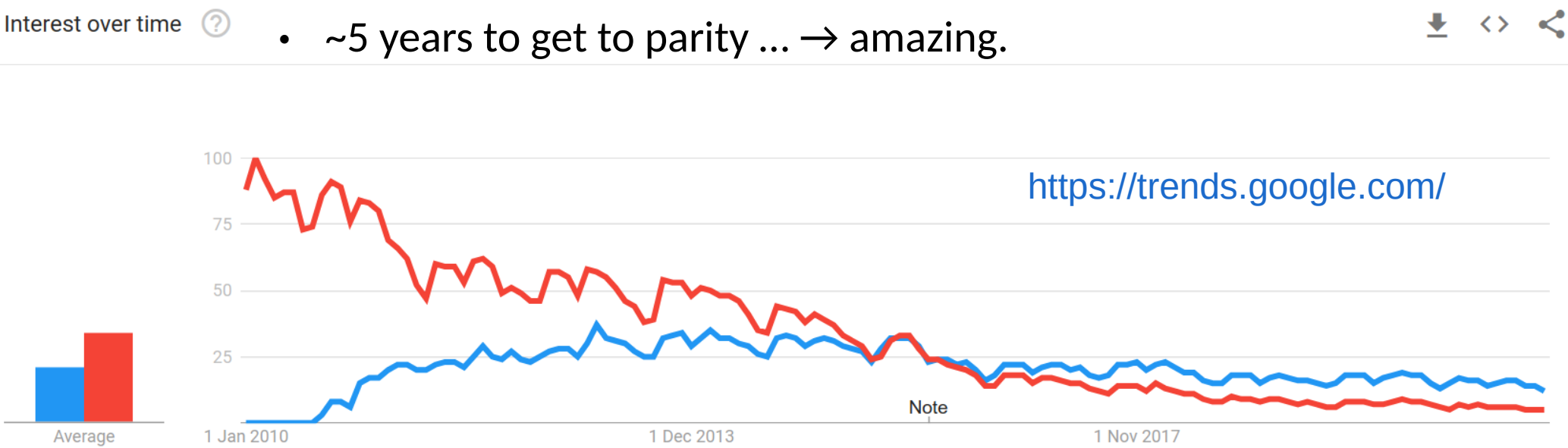
- Internationalization - i18n
- Accessibility - a11y
- security
- testing, validation, CI
- maintainability: cable-stay vs. suspension bridges.
- performance
- scalability
- manageability
- commercial: what can we afford ?
- marketing/sales: what can we sell ?

Skill #7 – Marketing ...

If we build it – they will come …

If we brand it – they will keep coming

- LibreOffice – vastly superior feature, function, interop etc.
- OpenOffice – still ~50,000 downloads per day avg.
  - Same price, different name: just brand mindshare…
  - ~5 years to get to parity … → amazing.



Skill #8 – Ignore programming language bigotry

My programming language is better !

Which programming language should I learn

- The wrong question → learn several:
  - At least one of: interpreted scripting + a compiled language + C/assembler

Programming languages are ~utterly banal

- None of them is going to give you the 10x productivity or performance improvement they claim.

Do not create a new general purpose programming language

- Whatever you do**[!]** - this just drives bifurcation from the bottom up
- By the time you have a working debugger – it will be obsolete.

A few I've worked with:

BBC Basic, 6502 assembler, x86 assembler, ARM32 assembler, Quick Basic, PASCAL, Turbo-Pascal, GNU Pascal, C, C++, Java, Visual Basic / VBA, C#, bash, perl, python, PHP, Javascript, SQL, m5, awk

various DSLs eg. yacc/bison, IDL(s), OpenCL, GLSL, AHDL, VHDL, YCP, ColdFusion

Some small patches of mine in the gcc compiler collection: C, C++

My business partner's advice: *"stay away from Haskell and Javascript"*

The world is rapidly filling with junk code, written in defunct languages

- LOC in the world grows very substantially: **bugs/1000 lines is ~constant at ~5**
- *Speaking with IEEE Spectrum, Technical University professor Manfred Broy explained that a **modern luxury vehicle** "probably contains close to **100 million lines** of software code" all of which is processed by up to 100 microprocessors networked throughout the car.* *Motor Authority*

But my language is more efficient / pretty!

Languages evolve (or die) – they all steal ideas from each other

As a Teen – (convinced I was God's gift to programming)

- I wrote computer games in assembler – on a 486
  - *"If it's not written in assembler, it's amazingly inefficient"* …
- Then I saw the assembler that gcc produced from C code
  - Wow! amazing register allocation, selected instructions I'd not heard of before … did a better job.
- Then I worked on real-time video editing systems in PASCAL [!]

Ultimately the Intel CPU itself is written in TCL[!] - huh.

- Lets get over the language thing.

For those that care see lang rankings

Skill #9 – Ignore (or understand) the Hype Cycle

The technology industry is a very odd place:

"The only way I can understand the computer industry: the computer industry is the only industry that's more fashion driven than women's fashion"

"What the Hell is Cloud Computing ?" - Larry Ellison

Get a good grip on simple concepts:

Object Orientation – tie a function pointer table to your data

Web 2.0 / AJAX – XMLHttpRequest – a single async web request API

Virtualization / virtual machines: ~8 CPU instructions 'VMRUN'++

Containers: 'chroot' syscall, namespaces + some scripts.

Kubernetes – some management / orchestration (written in 'go')

Skill #10 – Working on Free & Open Source software

Open Source: skill building par excellence

People Skills #1

- Contribute to an existing project with a team.

Self Motivation #2

- you don't **have to** work on FLOSS

Love Simplicity #3

- mentors / reviewers will notice your baroque solutions pretty fast.

Technique #4

- pick a big code-base so you have to learn to learn

Extreme focus & sustained attention #5

- extreme focus & sustained attention
- you really have to do that yourself.

Compromise & Collaborate #6

- Your beautiful code needs changing to get in!?

Marketing #7

- You'll have something to write about on your blog^Wtwitter^Winstagram^Wnext-IRC-clone

Learn new languages #8

- Lots of languages for everyone out there.
- ~all new languages are open-source anyway.

Ignore the hype-cycle #9

- See how little is under-the hood

Work on FOSS #10

- **Do Not** start your own FLOSS project.


Making
Open
Source
ROCK

Why not get involved with LibreOffice or COOL

LibreOffice features:

http://www.libreoffice.org/

Code

https://git.libreoffice.org/core

Get involved:

- https://www.libreoffice.org/community/get-involved/
- C++, python, l10n and more
- https://translations.documentfoundation.org/

Collabora Online (COOL)

https://collaboraonline.github.io/

Easy Hacks:

https://github.com/CollaboraOnline/online

- C++, Javascript, CSS, Design ...

https://hosted.weblate.org/projects/collabora-online/

Forum:

https://forum.collaboraonline.com/

Weekly public dev. Meetings: Thursday

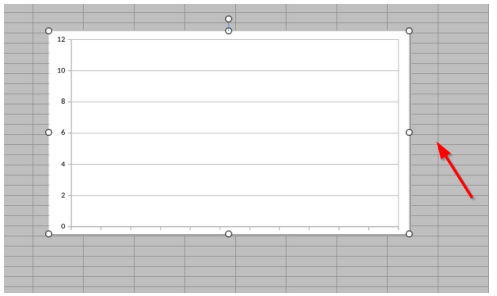
Nick’s experience

My experience at Collabora

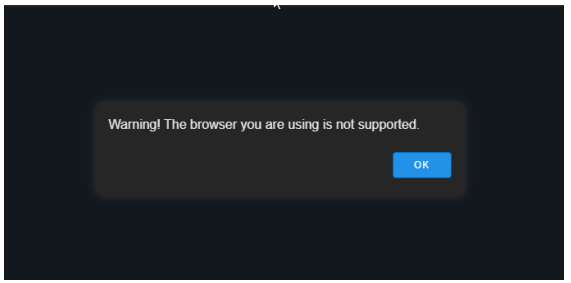
- Former Hills Road Sixth Form student (2020-2022)
- Started August 2021
- Paired with a mentor
- Given real tasks/bugs to fix → work until I get stuck → work some more → ask for help/tips → push fix to GitHub and see it merged

Some of what I've worked on

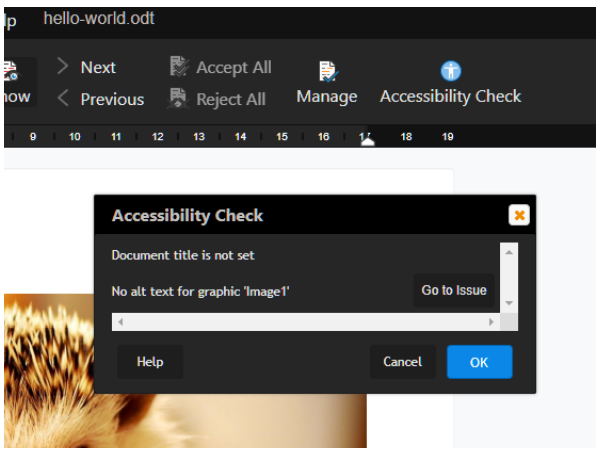
- **Optimizing dark overlay**



- **Warning users using unsupported browsers**



- **Accessibility Checker**

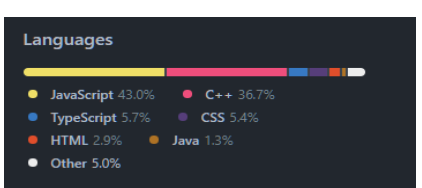


Learning the codebase

- Collabora has a huge codebase which is evolving daily
- Often come along commits/lines of code that are older than me

```
if(pSdrView->AreObjectsMarked())      Rüdiger Timm, 18 years ago • INTEGRATION: CWS aw013 (1.14.52); FILE MERGED …
```

- Initially overwhelming but the codebase is well documented and such a large codebase can be an advantage.
- Codebase streches over several design patterns, implementations, and languages.
  - Factory, Abstract factory, Command, State, Chain of responsibility



Summary of my time so far at Collabora

- Learned how open source software is maintained and software development in the real world
- Met new people and learned from them
- Furthered my passion for programming and a desire to have a career in it
- Improved my technical knowledge of the industry
- Made contributions to open source software
- **Learned many transferable skills:**
  - How to take practical notes in software engineering
  - Debugging & problem solving
  - Planning and allocating time between work/school/life

Postscript:
Beware.

Wealth is not a good goal [!]

Bernard Arnault & Family - **$186.3** billion

- Louis Vuitton ... (fashion)

Jeff Bezos - **$186** billion. ...

- Amazon: **degree** - electrical engineering & computer science

Elon Musk - **$147** billion. ...

- Paypal: physics / economics
- self taught programmer

  "I like C because it avoids class warfare"

Bill Gates - **$126** billion. ...

- pre-law, maths, computer-science

Mark Zuckerberg - **$115** billion. ...

- Psychology and Computer Science

"There is an elegance to writing code that I miss,"

Warren Buffet - **$109** billion

- Business Administration / Economics

Larry Ellison - **$102** billion

- Oracle: self taught programmer

Larry Page - **$100** billion.

- Google: BSC computer engineering

Sergey Brin - **$97.1** billion

- Google: BSC computer science

Amancio Ortega - **$89** billion

- Zara fashions

some quite noxious people in this list.

2021 - top 10 billionaires: Business Insider

The *love* of money is the root of all kinds of evil.

We work in an exciting industry

- Opportunity for all beyond belief.
- Work hard & you will do well & have an exciting time.

Last key skill: remember the real world

- Don’t disappear into your bubble / ignore your family
- Have a reasonable study/life balance.

"Our inventions are wont to be pretty toys, which distract our attention from serious things. They are but improved means to an unimproved end" - Thoureau.

Thank you!

By Michael Meeks
@mmeeks @CollaboraOffice
CollaboraOffice.com
CollaboraOffice.com/CODE
michael.meeks@collabora.com

Open Source, Open First

Oh, that my words were recorded, that they were written on a scroll, that they were inscribed with an iron tool on lead, or engraved in rock for ever! I know that my Redeemer lives, and that in the end he will stand upon the earth. And though this body has been destroyed yet in my flesh I will see God, I myself will see him, with my own eyes - I and not another. How my heart yearns within me. - Job 19: 23-27