Solving arbitrary problems from a standing start

Or – Real World Software Engineering technique – to improve your effectiveness.

Michael Meeks

CEO

michael.meeks@collabora.com

"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

Collabora Online

Solving hard problems:

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)

Means working hard ...

Still: Technique helps

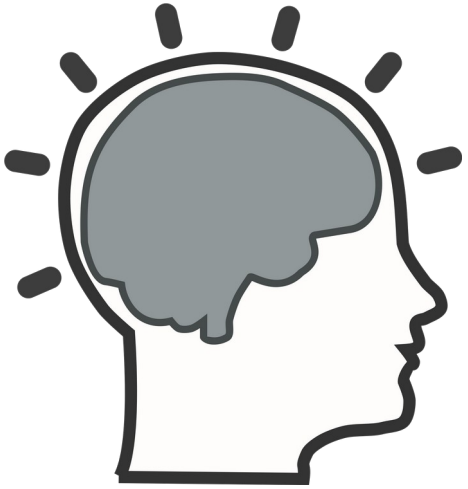
Something anyone can
learn – but no-one wants to.

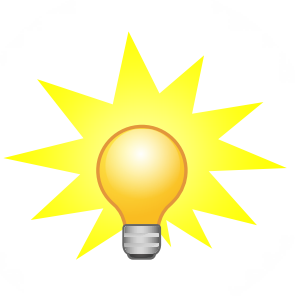
Abandon intuition & trying – it only goes so far

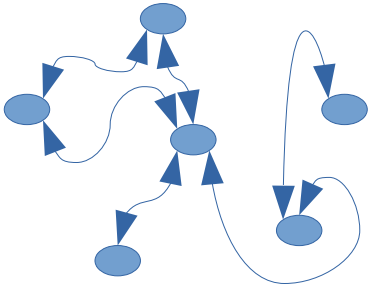
Computers behave deterministically – how hard can it be ?

Many Computer Science Graduates

- Excellent at small problems, creating small new programs







Intuition as a approach:

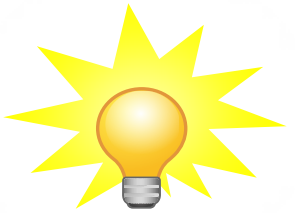
Technique tends to be either

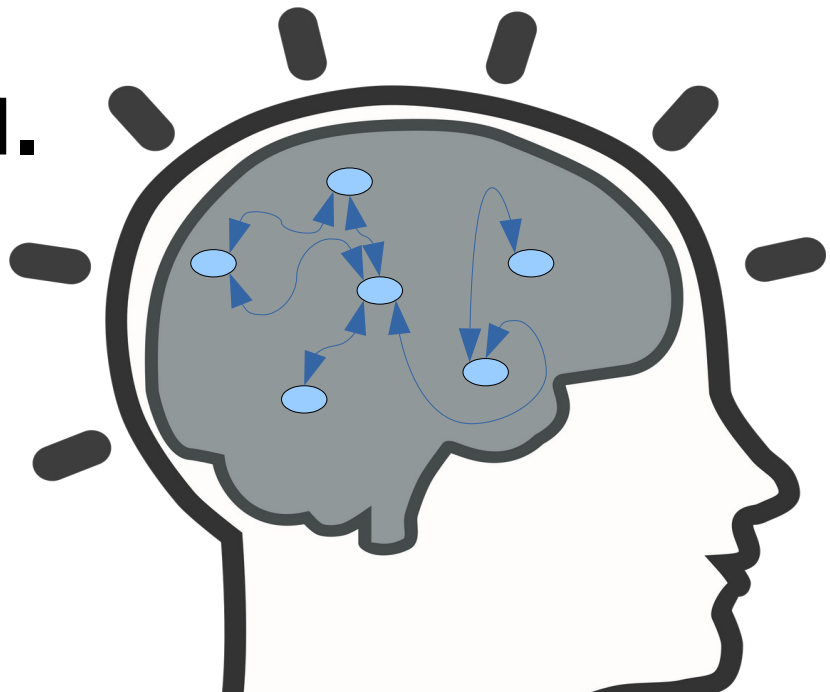
- ‘intuition’ or understanding the
- Whole problem:

fitting it all in your mind.

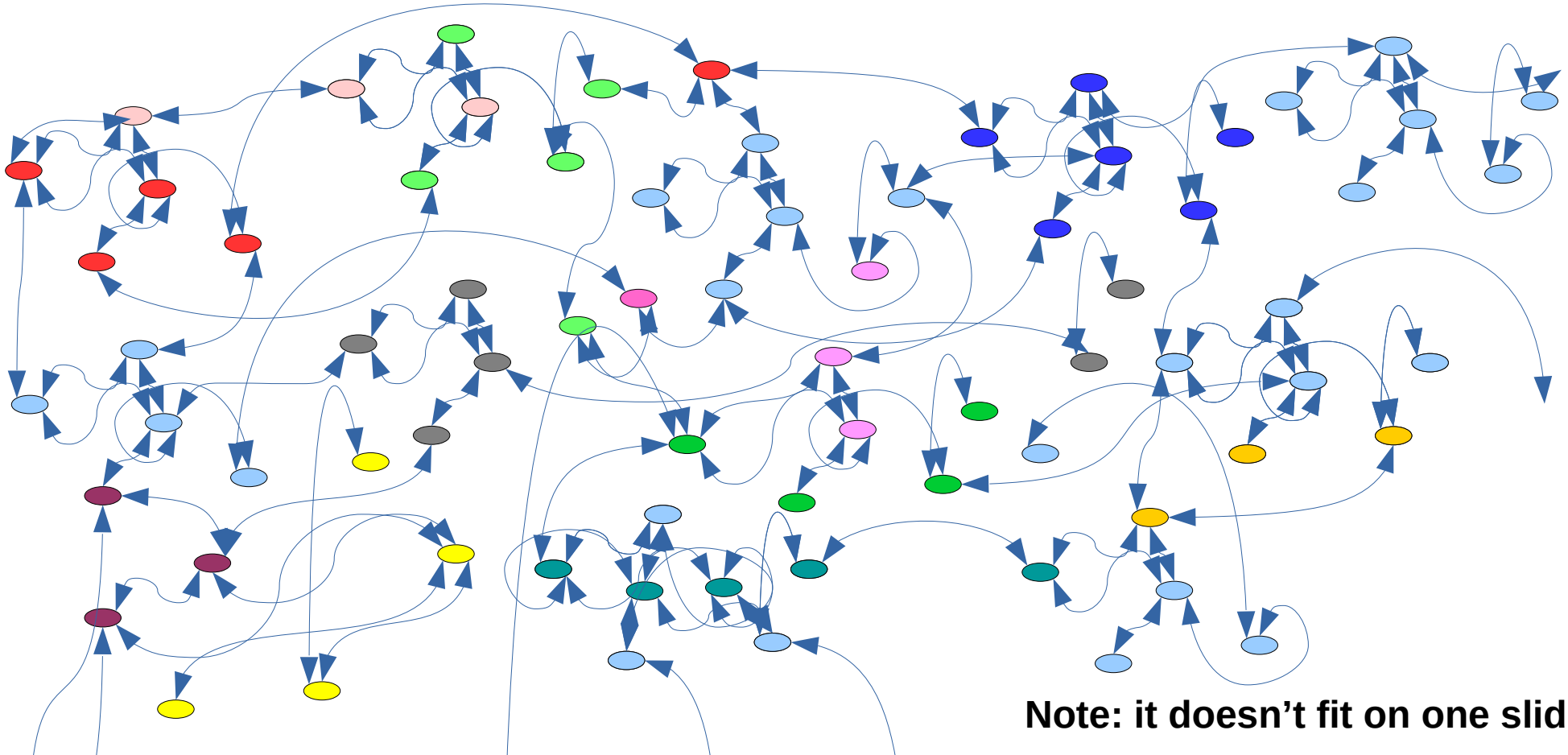
“If I have two kinds of fish, and they are in the ratio 3:2 big to small, and there are 4 small fish, how many do I have ?”

“It’s obvious” → solution ...





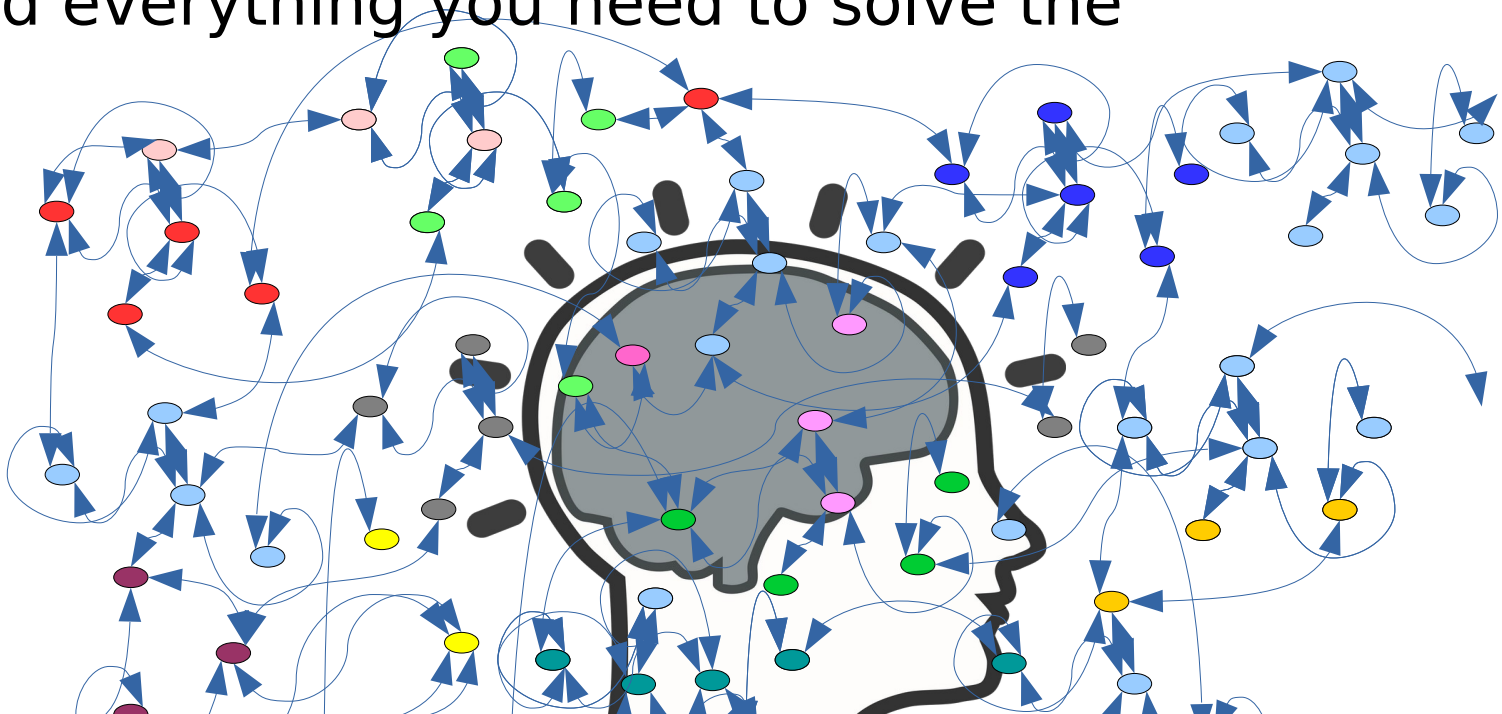
A LibreOffice problem looks like:



Note: it doesn’t fit on one slide

Problem domain >> head size:

- 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.



So what do we do ?

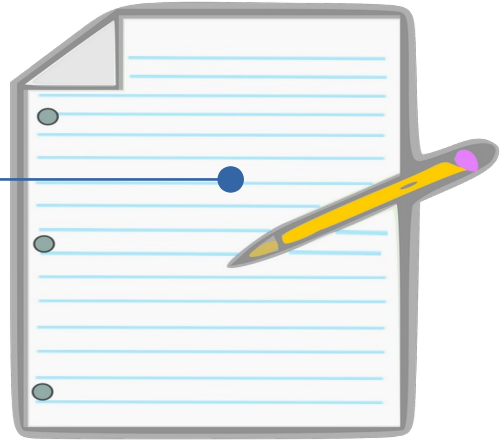
New Skill - can only be learned by tackling problems that you cannot solve without it.
The pre-eminent engineering skill.

Reductionism is your friend

"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 ?

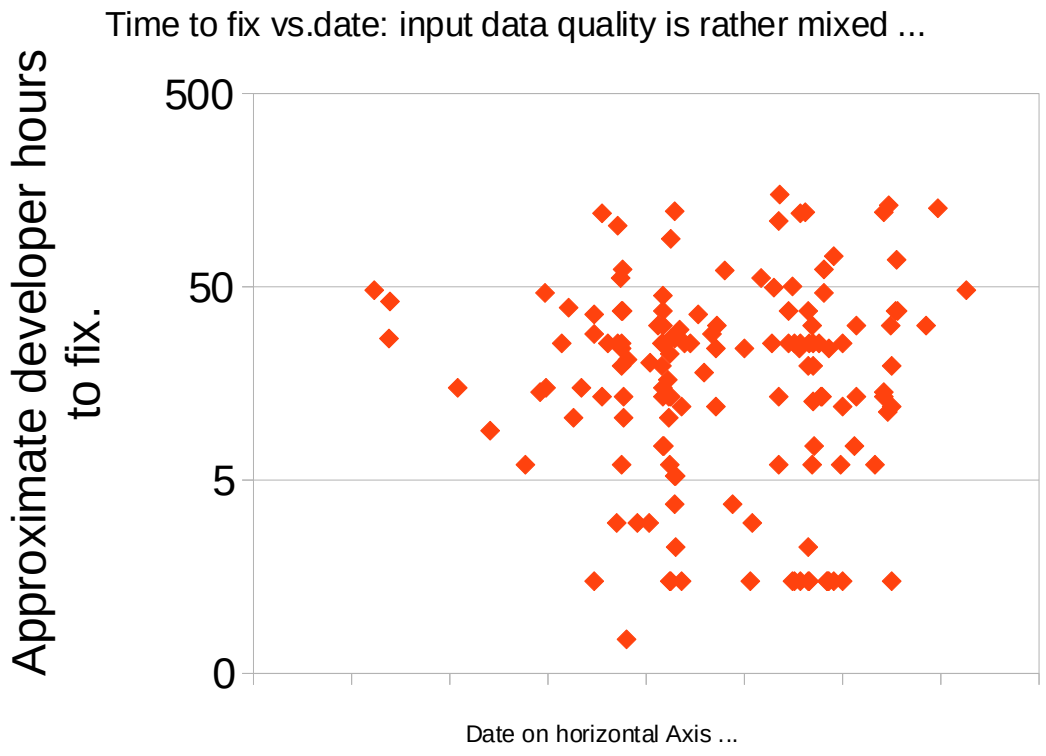
- The magic tool:
  - Break the problem into small steps
  - write them down
  - check them.



Why write things down ?

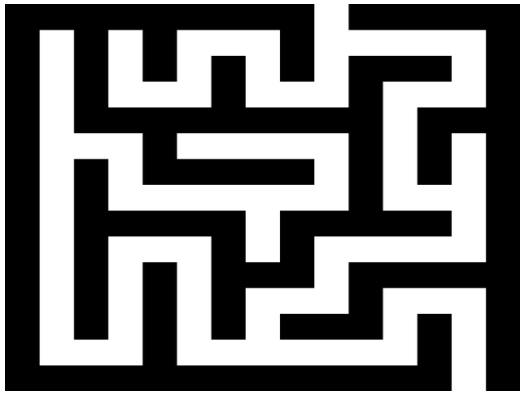
Some time-to fix data (by expert developers):

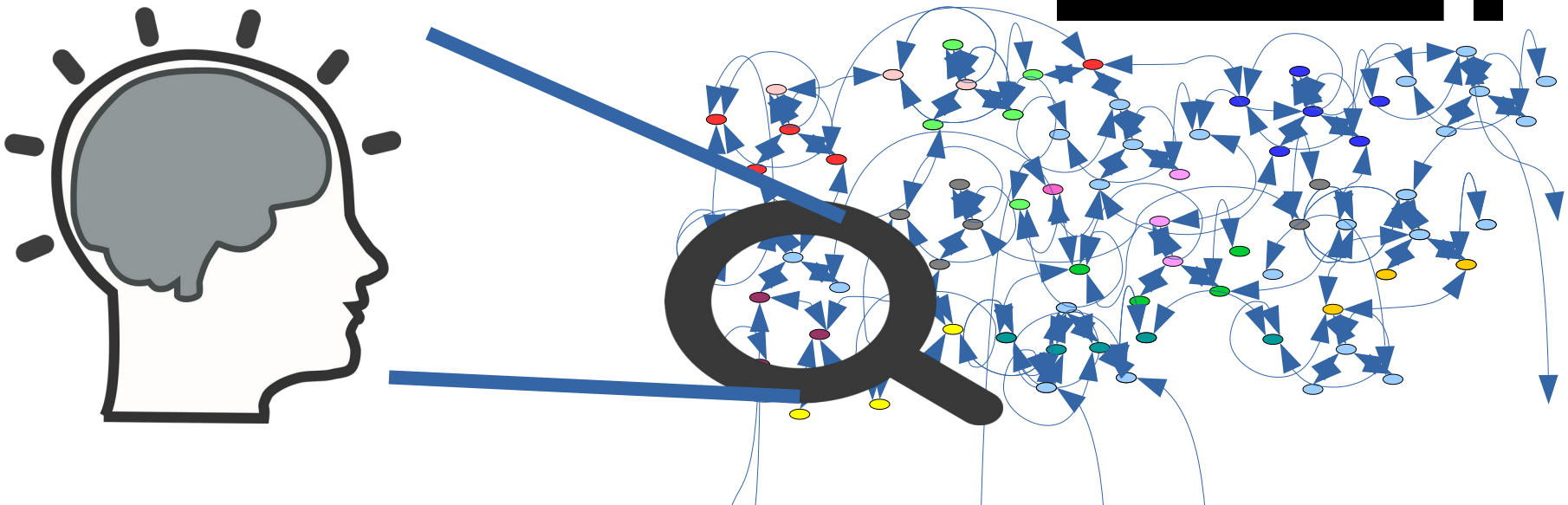
- notice Log scale.
- Average one week to fix.
- Can you hold and make progress in your head for a week ?



Hunting in the code graph

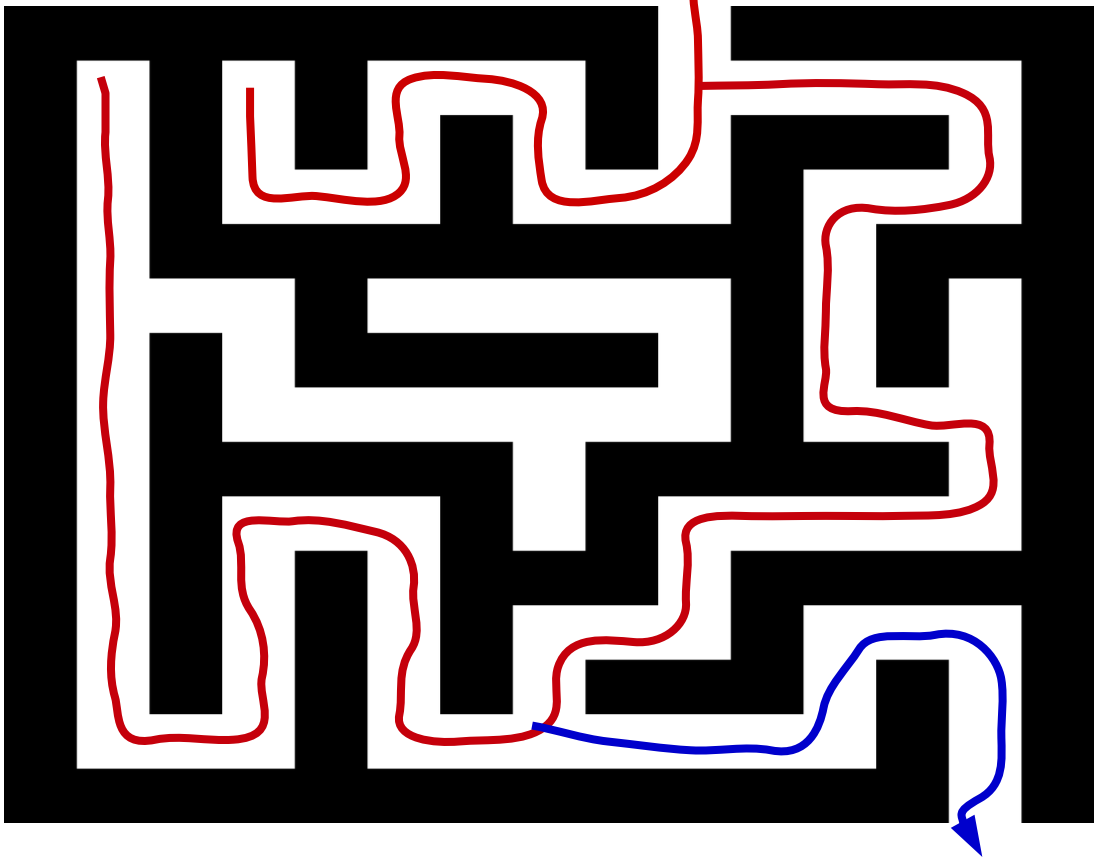
- As you walk the code, in small bits – you will hit dead ends – be patient.





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.



What I expect in your log

- Cargo cult useful looking code.
  - If you're following a stack-trace:
    - Highlight the trace with relevant code:
    - Cleanout all exceptional / non-taken paths for this case.
- Code pointers → where the relevant source code is ...
- Include printf / log output proving things
- Detail your assumptions:
  - *"XYZ cannot happen → it must do ABC."*
    - You can re-examine these later when you find that it does not do ABC
- Energetically Enumerate Emergent Axioms (or something)
- If you are asking a mentor for help:
  - Prove you did your leg-work by sharing your log.

What I don’t expect in your log

- Not a neat essay …
  - This is a collection and grab-bag of bits of code.
- Not for publication
  - I’ll show you some of mine but they don’t end up in slide-decks & on the web.
- Not perfect:
  - It is the scrabbly “working” in your proof.
  - Errors are expected.
- Too much work:
  - You -cannot- read all code-paths →
    - document your assumptions: eg.
    - *“calling bool OUString::endsWith() cannot affect Bitmap rendering behavior”*

Self Rubber-Ducking:

Thinking is not speaking.

- But language can help.

Explain your problem to a rubber duck in the bath:

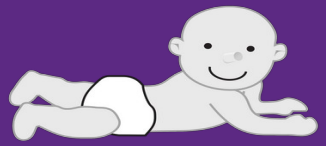
- Bear in mind the Duck is in fact a brilliant engineer & is listening hard.
- Watch your understanding expand as you wrestle with the problems.

Re-reading your log serves this purpose.

Hardware engineers: fault tree analysis

Towards being the ideal mentee: and learning fast

Three approaches:

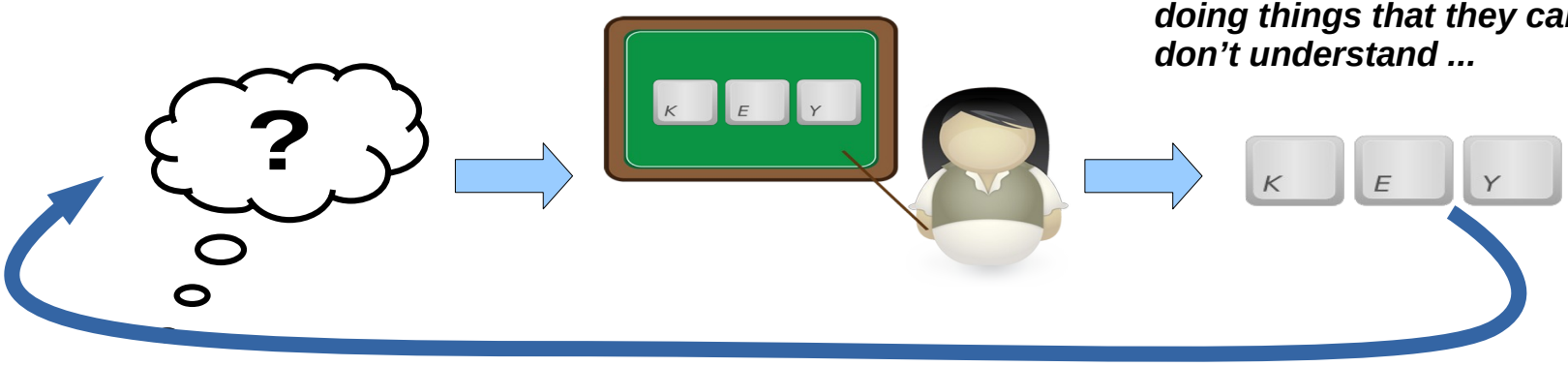


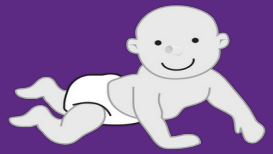
The Chatty Anti-Pattern

- The remote typing, mentor loop.
- For bonus points – interact with several mentors ... gerrit review is an ask

Let me google that for you
lmgtfy.com/
For all those people who find it more convenient to bother you with their question rather than google it for themselves.

This is also how managers irritate engineers into doing things that they can't do themselves, and don't understand ...





The Stalling Anti-Pattern

- Not a good place to end up.
- Comes to us all sometimes.
- Good to seek re-assurance
  - Best to do so with a concrete plan.

Trough of doubt:
Indecision / Can it really be this bad ? / Does it really take this much effort ? Surely there is an easier way ?

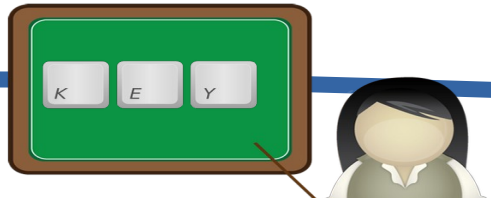
STOP

The competent developer

Ideal:

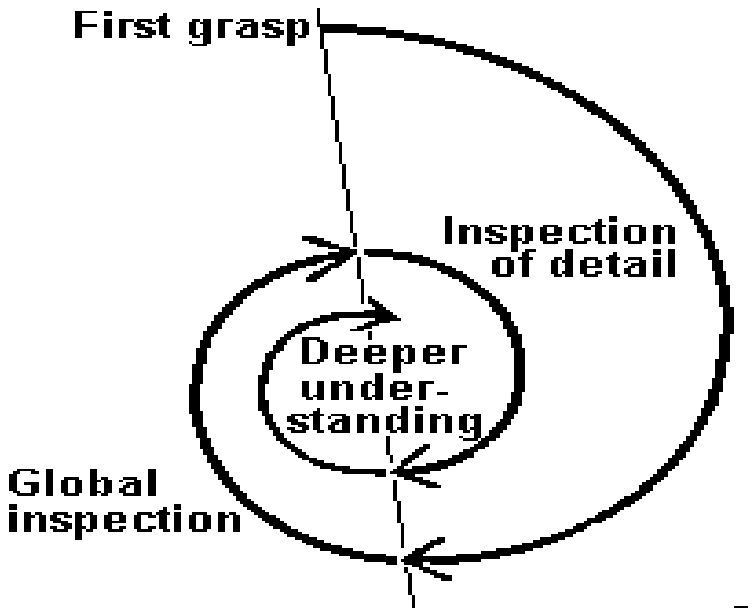
Code reading
Google
Stack Overflow
Using the debugger
Printf logging
Trace analysis
Tweak code test
Check for anti-patterns then ...

Please bear in mind: that for any non-trivial question, a mentor has no ‘magic’ answer – they often need to do this themselves



The Hermenutical Spiral

- **Hermenutics:** the Philosophy and Methodology of text (in our case code) interpretation.
- It is necessarily iterative:
  - Rinse and repeat:
    - Build referents
    - Re-apply this
- NB. *Please* also re-apply your new understanding to the advice and input given by a mentor
- Go back and re-read their words before asking ...



Skill #4 – Technique

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

The pre-eminent engineering skill.

Conclusions:

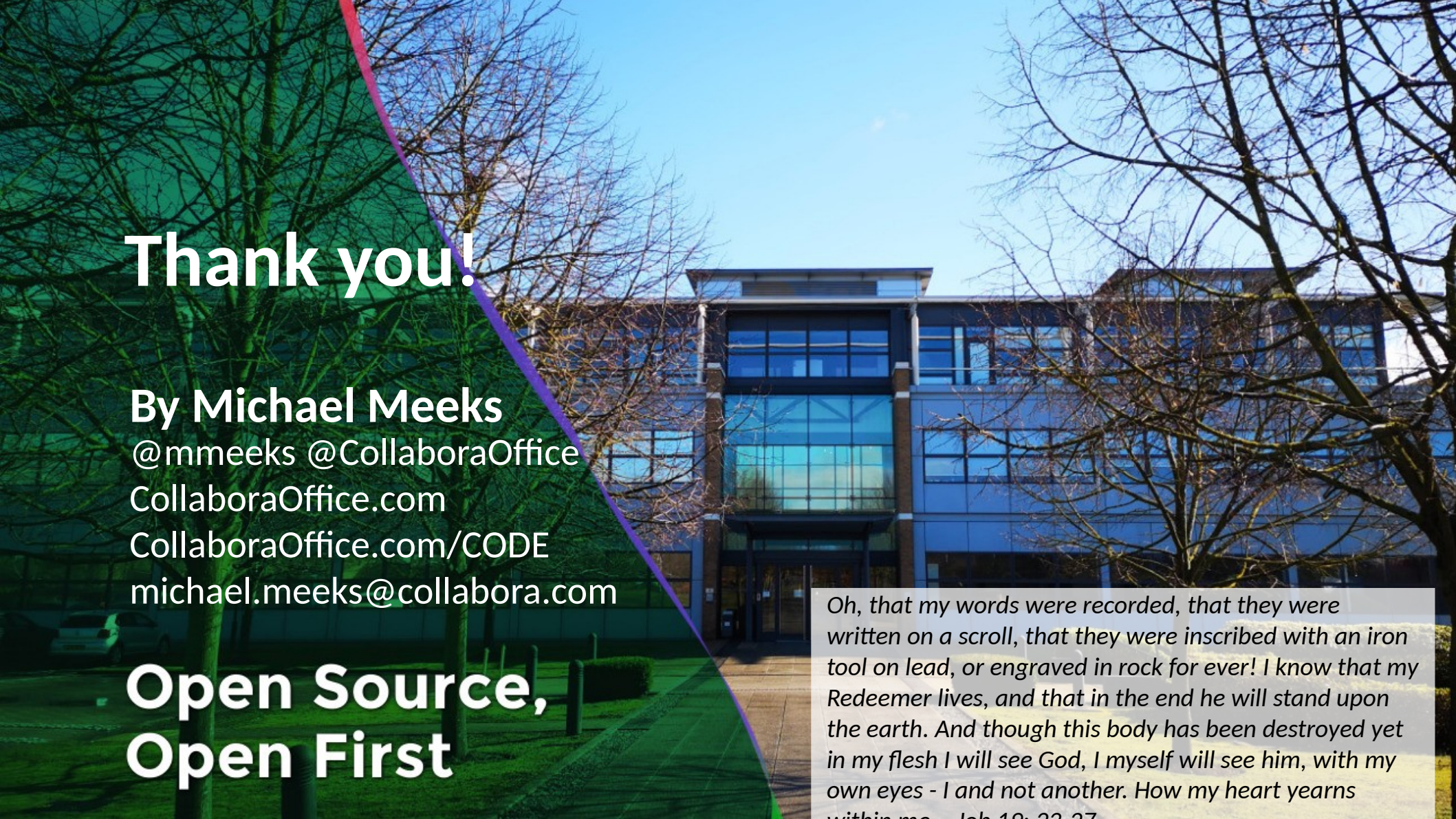
- It is possible to break down arbitrarily awful problems and solve them
- If you can work on LibreOffice and defeat it you can work effectively on any large code-base
- This is the pre-eminent engineering skill
- It is not easy, but you can do it and well.
- Don’t be overcome by evil [code], but overcome evil with good [code].
- **Don’t assume it is easy and cheap for your mentor to answer a question.**

Making Open Source ROCK

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