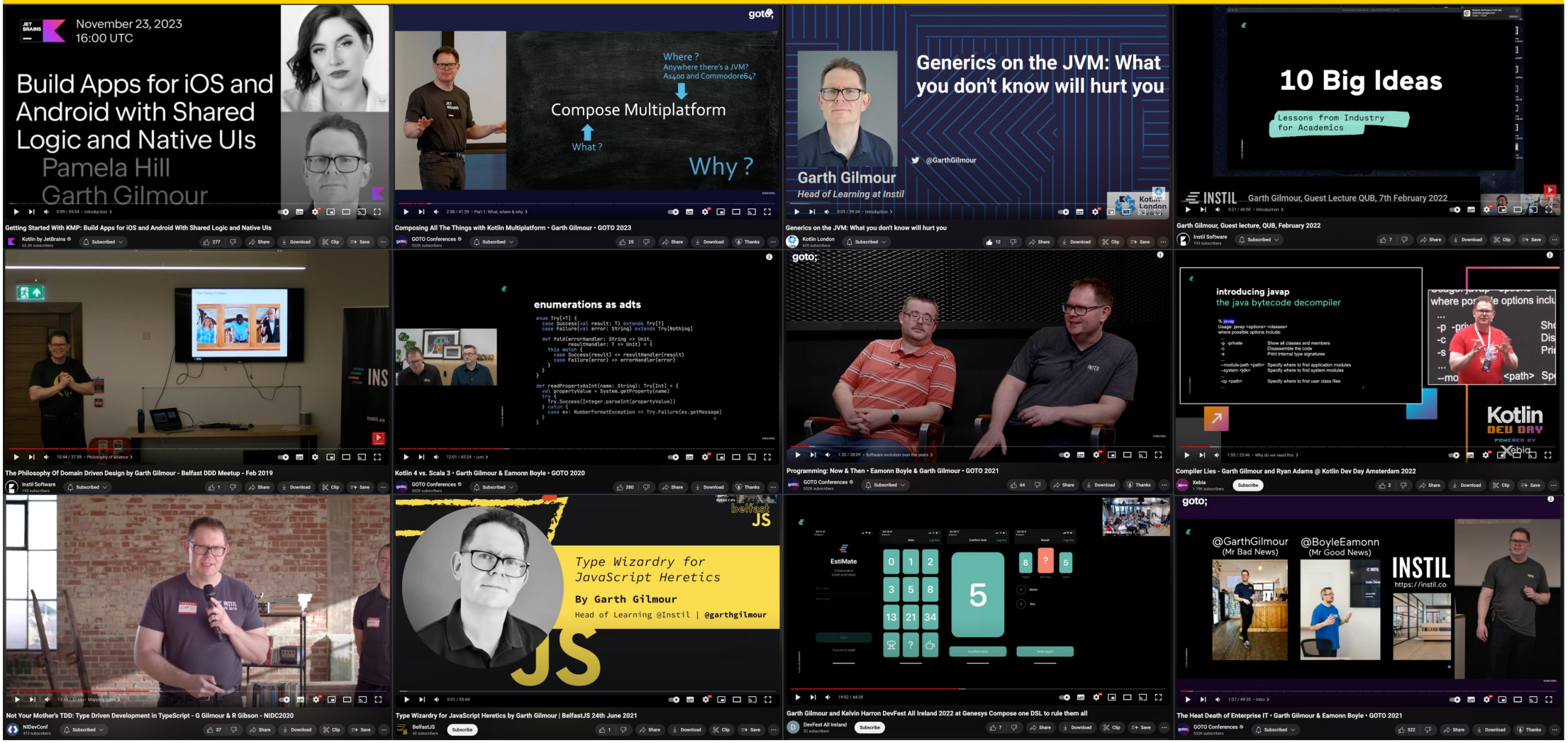
Lessons from 25 years in post-tertiary education

CAS NI Summer Mini Conference – June 2026

Garth Gilmour (Garth.Gilmour@LibertyMutual.com)

About Me (Learning Consultant at Liberty IT)



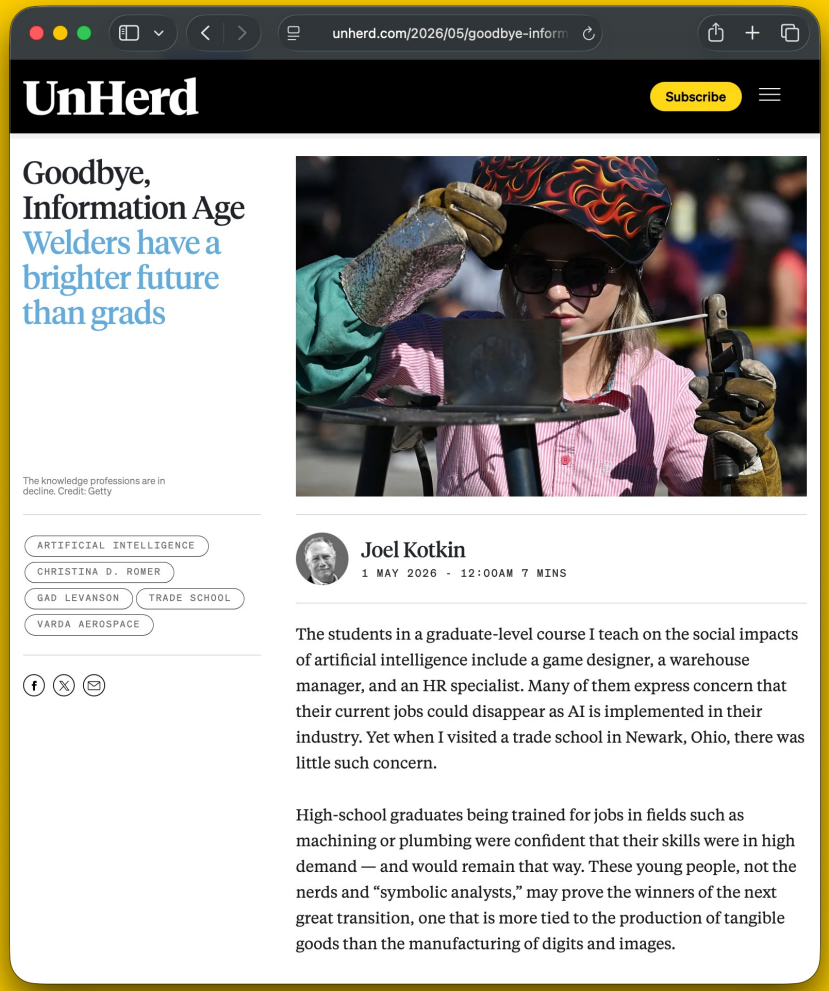
Agenda

- Take friction points between CompSci and SoftEng
  - Which historically have been difficult for academics to overcome
- Examine them in a world with GenAI
  - After a short summary of the journey that we are all on
- Do the AI based tools make things better or worse?
  - If better, how do we make the most of the opportunities?
  - If worse, how do we mitigate the damage being caused?

1st version written for lecturers, but the content is applicable to educators in general

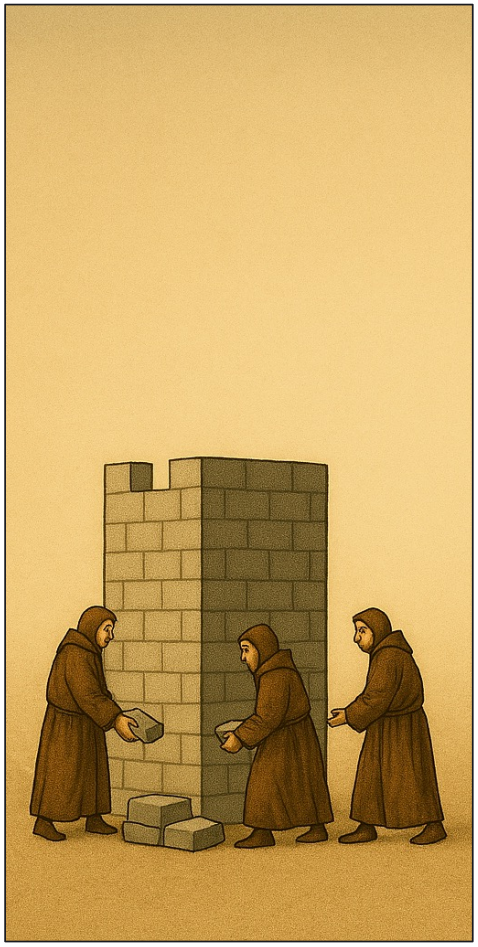
“I’m so glad I didn’t choose Software Engineering. Those jobs are gone.”

Source: A student I overheard on the Larne Line train.



https://unherd.com/2026/05/goodbye-information-age/

Part 1 – The Great Debate



Programming in an undergraduate CS curriculum

Bjarne Stroustrup
Texas A&M University
bs@cs.tamu.edu
www.research.att.com/~bs

Abstract

This note argues for a fairly classical undergraduate computer science (CS) curriculum where "software" (programming and related topics) takes a bigger role than is often the case. The discussion is based partly on experience with an undergraduate curriculum change at Texas A&M University and with developing a new freshman programming course. That freshman course is the central topic of this note. Based on industrial experience, it is argued that the primary aim of a university education in the area of "software" is to be a foundation for professional work. The primary design criterion for the freshman (first year) programming course is to make it a good start at that. Caveat: the opinions expressed about the needed improvements of and directions for software education is based on personal experience rather than hard data.

1 Introduction: Problems

My perspective is that of an industrial researcher and manager (24 years at Bell Labs; 7 of those as a department head) who has now spent 6 years in academia. I see a mismatch between what universities produce and what industry needs (not just what industry says it needs). When I say "industry" I obviously simplify, but I base my simplification on talks with dozens of representatives of (primarily, but not exclusively, US) industry a year over the last 30 years or so.

Industry wants computer science and computer engineering graduates to build systems consisting largely of software (at least initially in their careers). Many graduates have essentially no education or training in that outside their hobbyist activities. In particular, most see programming as a minimal effort to complete homework and rarely take a longer-term view involving use of their code by others and maintenance. Also, many students fail to connect what they learn in one class to what they learn in another. That way, you can (and often do) see students with high grades in algorithms, data structures, and software engineering hack solutions in an operating systems course with total disregard for data structures and algorithms, resulting in a poorly performing unmaintainable mess.

For many, "programming" has become a strange combination of unprincipled hacking and invoking other people's libraries (with only the vaguest idea of what's going on). The notions of "maintenance" and "code quality" are at best purely academic. Consequently, many in industry despair over the difficulty of finding graduates who understand "systems" and "can architect software."

This is not all or even primarily the fault of the students. The academic curriculum is crowded with topics of undisputed importance and in the resulting competition for time, practical issues and the development of time-consuming practical skills (such as design for testing) lose out to more classical academic subjects (such as mathematical analysis of algorithms) or fashionable research topics (such as subsurface luminosity). However, to remain an applied discipline – as it has been from its inception – computer science must emphasize software development and CS programs must allocate time for student skills to mature. If we don't, we are like a music department that does not require musicians to practice before a concert or an athletics department that "trains" its athletes primarily through lectures.

For the good of both industry and academia, we must do better.

2 What do we want?

A university education should lead to a well-rounded personality, provide a solid grounding in an academic field, and be a reasonable preparation for a job (or grad school entry). I consider a basic knowledge of history, art, math, and science (e.g., biology and physics) essential, so I'm not going to argue for more CS at the expense of a liberal education. I just wish the liberal arts curricula similarly went out of their way to avoid producing scientific ignoramuses.

The majority of CS graduates do not become professors, so I consider preparation for a career in industry crucial. I use the term "industry" in its broadest sense, including non-profit organizations and government. Industry does not want "scientists" in large numbers (I have heard repeated reports of recruiters denying interviews to "computer scientists" for

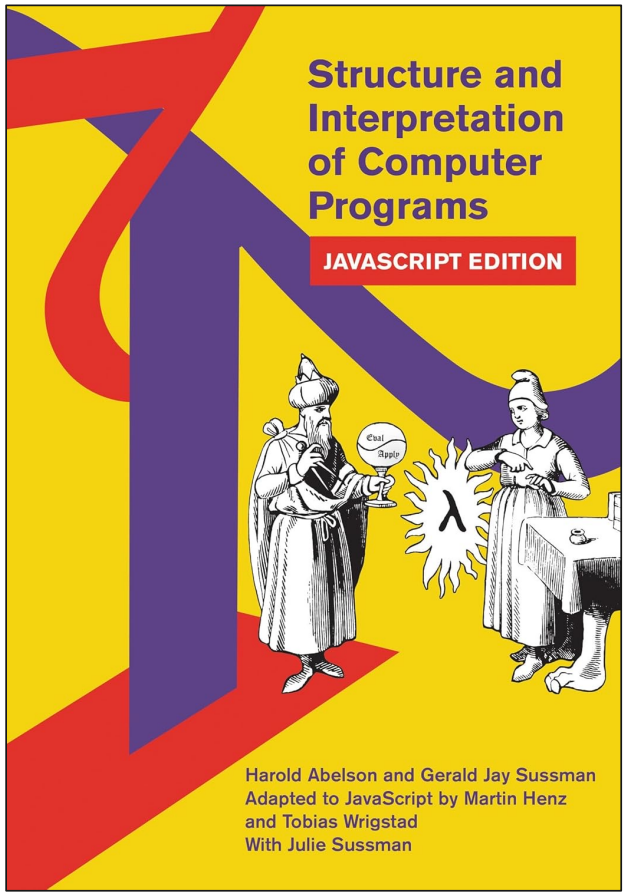
WCCCE '09 May 1-2, 2009, Vancouver, British Columbia, Canada.

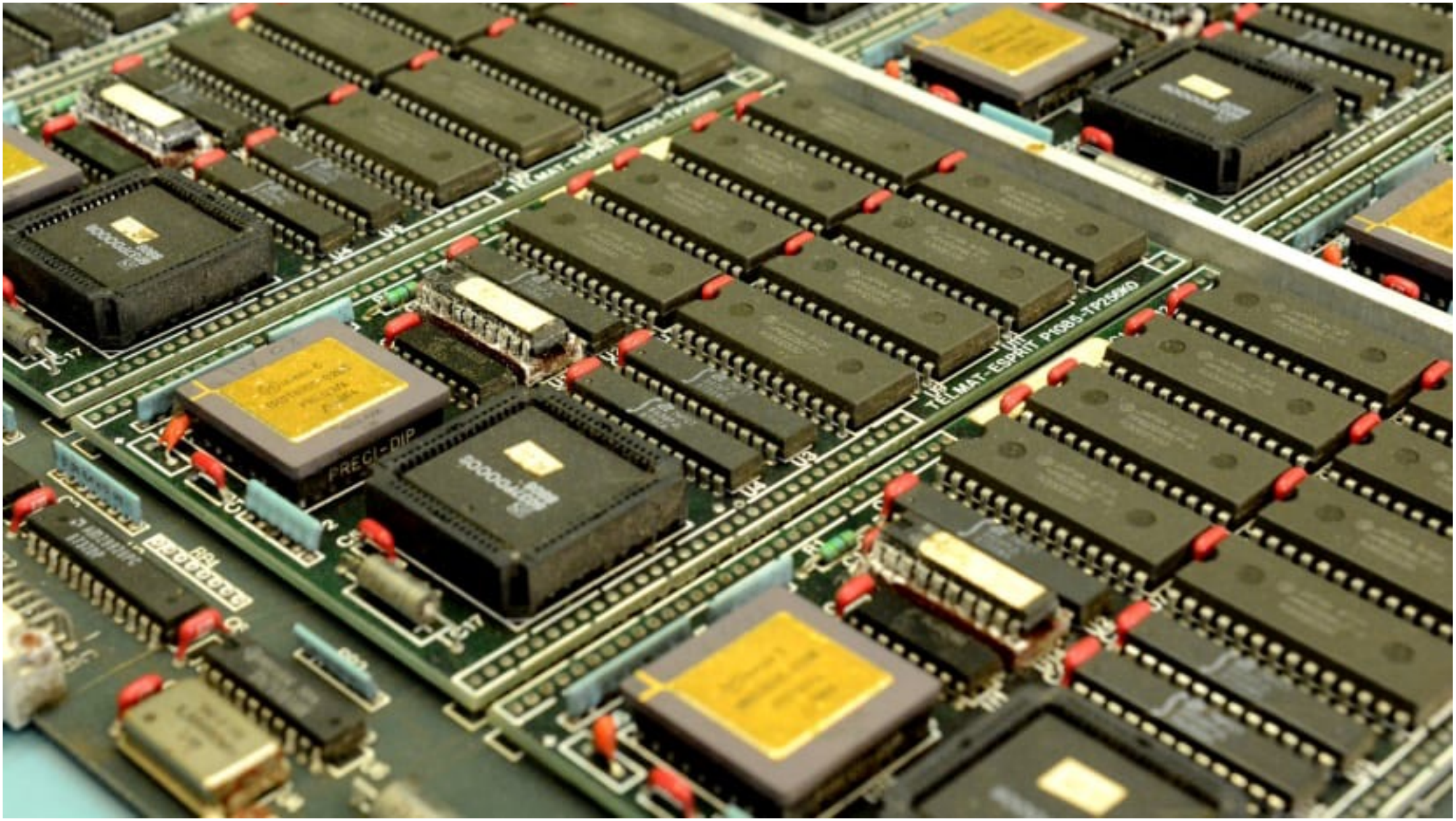
https://www.stroustrup.com/software.pdf

"The academic curriculum is crowded with topics of undisputed importance and in the resulting competition for time, practical issues and the development of time-consuming practical skills (such as design for testing) lose out to more classical academic subjects (such as mathematical analysis of algorithms) or fashionable research topics (such as subsurface luminosity). However, to remain an applied discipline – as it has been from its inception – computer science must emphasize software development and CS programs must allocate time for student skills to mature. If we don't, we are like a music department that does not require musicians to practice before a concert or an athletics department that "trains" its athletes primarily through lectures."

The Virtues of Education

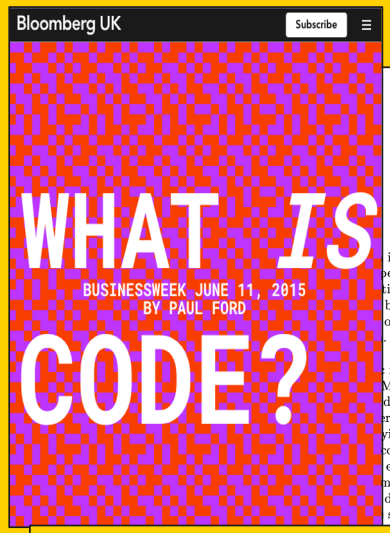
- Learning the fundamentals
  - Syntax trees, compilers, garbage collection
  - Data structures, algorithms, abstraction
  - Databases in all their varied forms
  - Theory of distributed systems
- Exploring how to design software
  - With the freedom to make mistakes
- Exposure to the ‘bleeding edge’
  - Stuff students will never see in Enterprise IT




PRECI-DIP

Traps Specific to Education

- Falling between two stools
- Not keeping pace with modernity
- Using obscure languages and tools
- Continuing to support obsolete platforms
- Omitting project work and version control
- Not communicating the realities of job roles

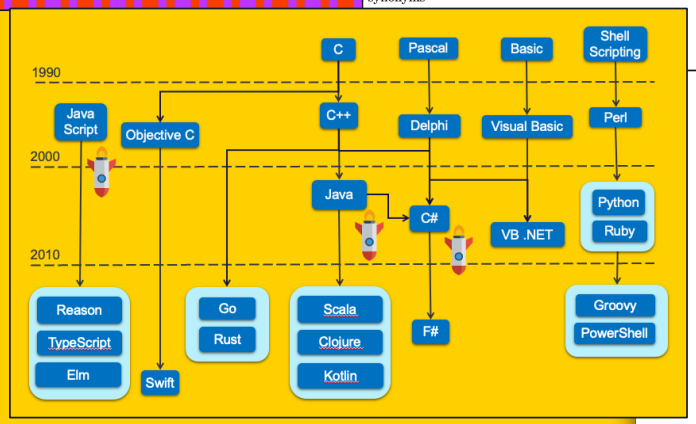


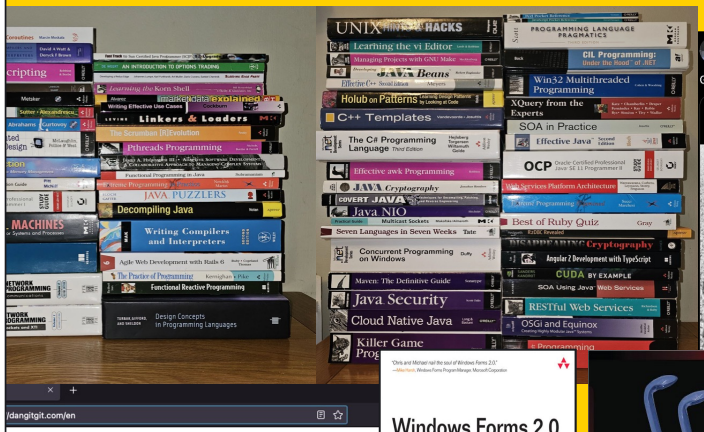
An Overview of Miranda

David Turner
Computing Laboratory
University of Kent
Canterbury CT2 7NF
ENGLAND

is an advanced functional programming system which runs under the
perating system[1]. The aim of the Miranda system is to provide a mod-
ional language, embedded in a convenient programming environment,
both for teaching and as a general purpose programming tool. The
of this short article is to give a brief overview of the main features of
. The topics we shall discuss, in order, are:

ideas
Miranda programming environment
ded equations and block structure
ern matching
ying and higher order functions
comprehensions
evaluation and infinite lists
morphic strong typing
defined types
synonyms





I committed and immediately realized o make one small change!

```
# make your change
git add . # or add individual files
git commit --amend --no-edit
# now your last commit contains that change!
# WARNING: never amend public commits
```

This usually happens to me if I commit, then run tests/linters... and ugh, I didn't put a space after an equals sign. You could also make the change as a new commit and then do `rebase -i` in order to squash them both together, but this is about a million times faster.

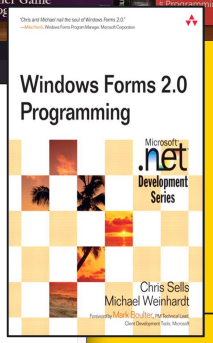
Warning: You should never amend commits that have been pushed up to a public/shared branch! Only amend commits that only exist in your local copy or you're gonna have a bad time.

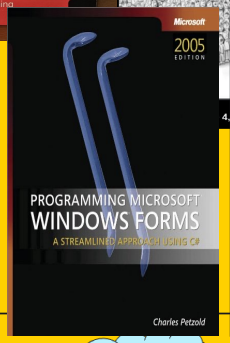
Dangit, I need to change the message on my last commit!

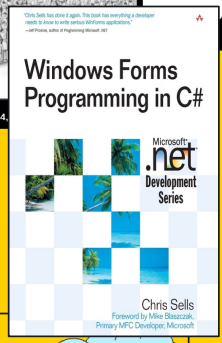
```
git commit --amend
# follow prompts to change the commit message
```

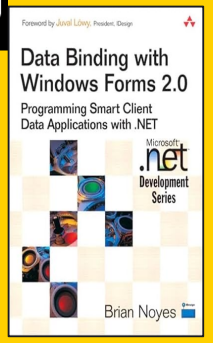
Stupid commit message formatting requirements.

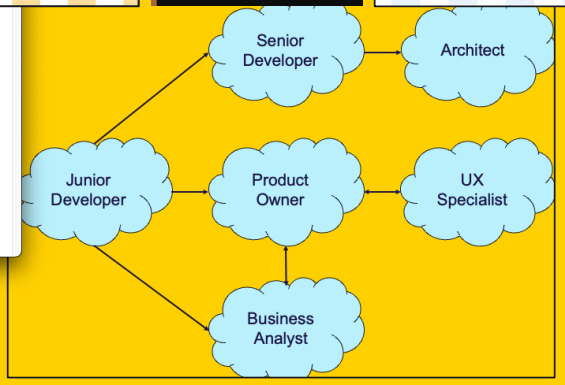
Dangit, I accidentally committed something to master that should have been on a brand new branch!











Are Educators Less Important Today?

- Technical documentation is routinely provided for free online
- Conference talks and academic lectures are available on YouTube
- On-line courses are available for any subject you care to name
- Non-academic engineering books are available for a modest cost

No! Educators Are More Important Today

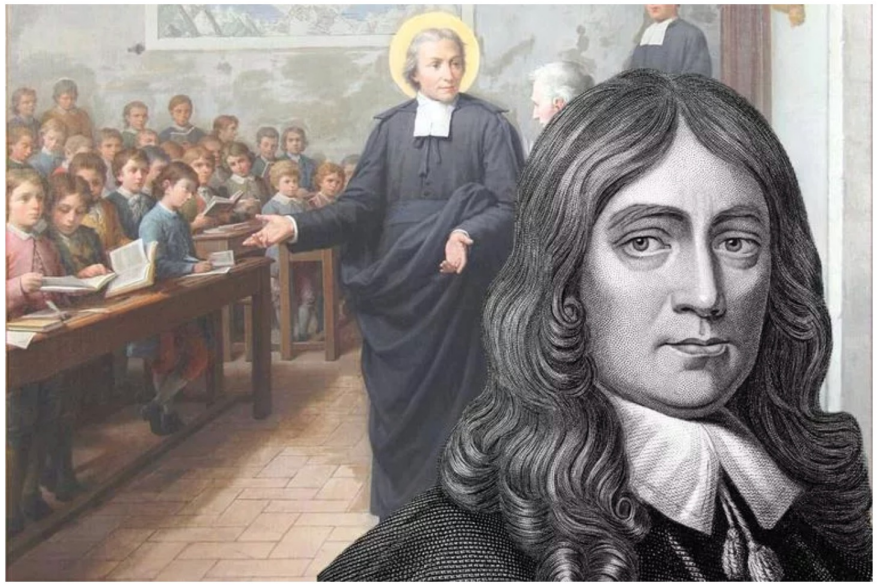
- This cornucopia means students need context and guidance
- There is enormous pressure to memorize rather than think
- Students need to be forced to learn how to engage and question
- Separating rhetoric from reality has never been more important

NATIONAL CATHOLIC REGISTER

Church & AI | Classic Books | Catholic Hubs | Death Issue

Repairing the Ruins: Why AI Can't Replace Education

COMMENTARY: In an era when AI can write anything, authentic education must go beyond the mere production of words.



O. Cook, "John Milton (1608-1674)," engraving from the 1800s. Background: St. Jean-Baptiste de La Salle, the patron saint of teachers, visits a school in Paris on Rue Princesse, after a work by Giovanni Gagliardi (1838–1924) created in 1901. (photo: Shutterstock / Wikimedia Commons)

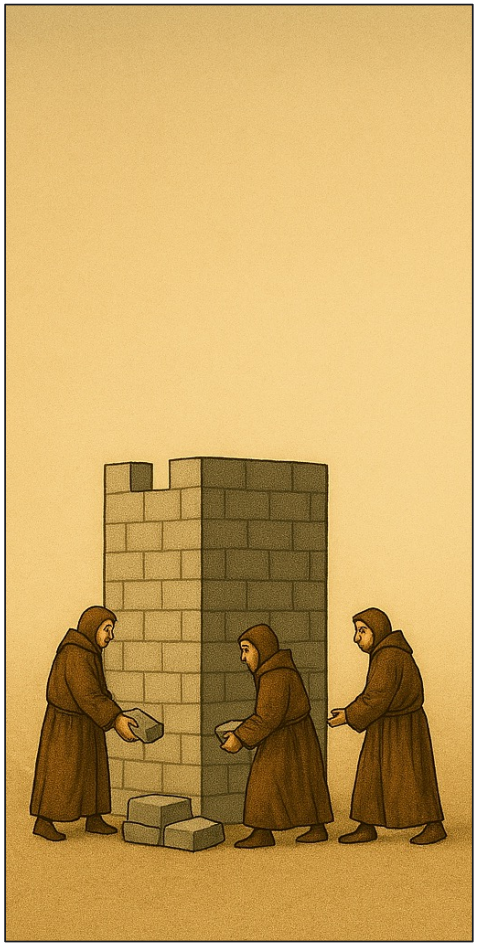
https://www.ncregister.com/commentaries/schnell-repairing-the-ruins

"The teacher's role accordingly becomes more important in the age of AI, not less. A real teacher is not merely a distributor of content. A real teacher is an experienced guide in inquiry: someone who knows what the student has not yet seen, what distinctions must be made, what confusion needs exposing, and what question should come next. The best classroom is not a transfer of information from one container to another. It is a living act of thought. That is why seminar, disputation, laboratory, tutorial and serious conversation retain their force even when information itself becomes cheap."

"This clarifies why certain acts cannot be delegated to machines without ceasing to occur at all. Attending carefully to a text, weighing conflicting evidence, judging whether a conclusion is warranted, taking responsibility for what one claims — these are not ancillary tasks. They are the work by which a mind is formed.

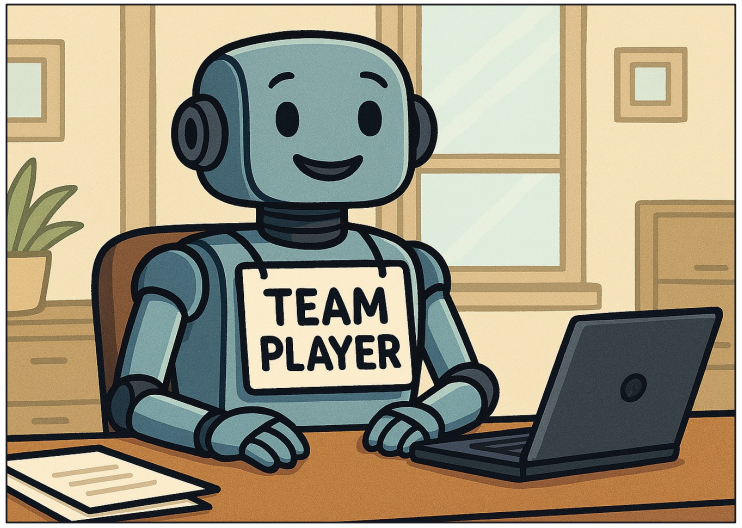
No machine can perform them in our place — not because machines lack processing power, but because these acts have no effect unless a person performs them. Their purpose is not to produce an output. It is to form the one who does them."

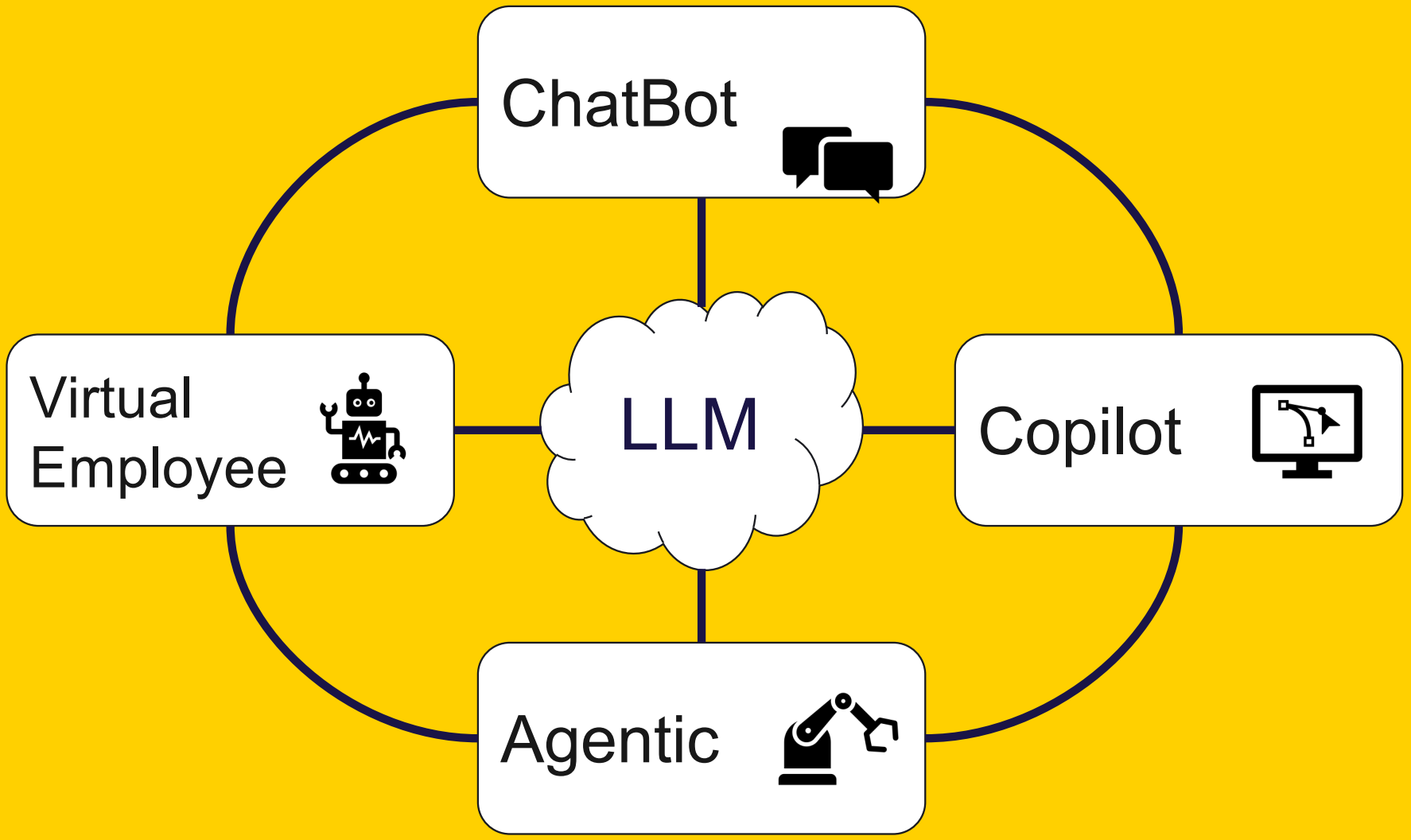
Part 1b – Our GenAI Journey



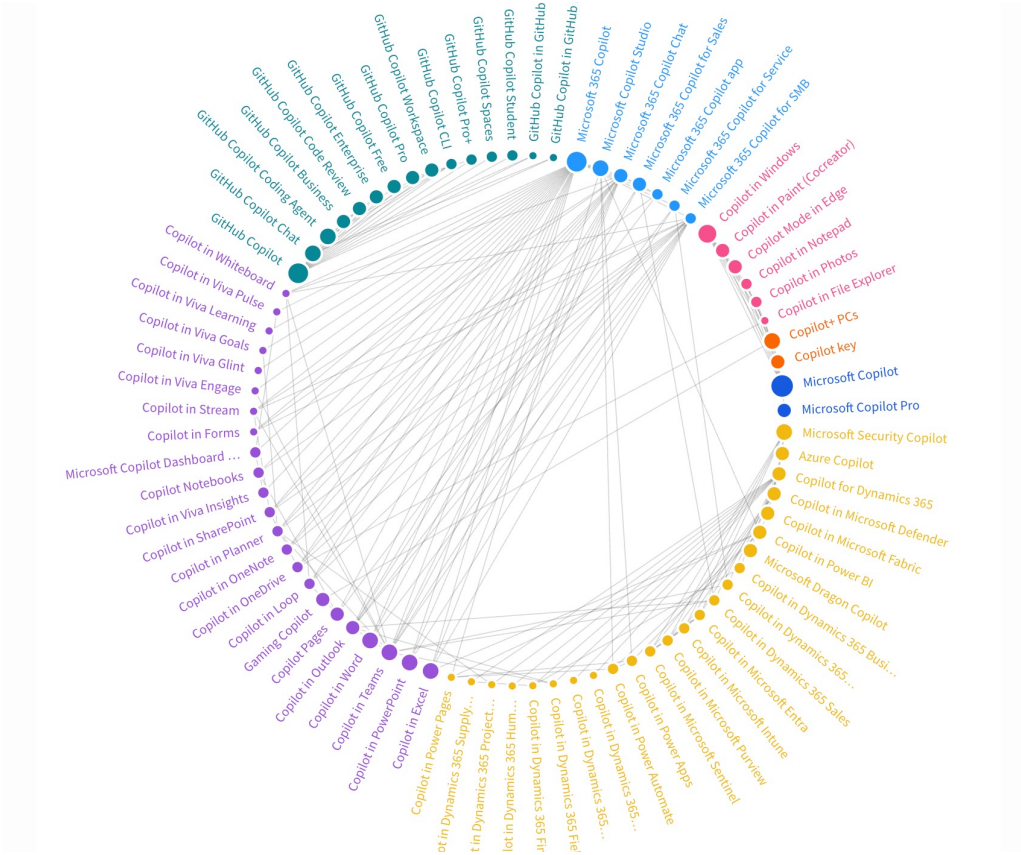
Our GenAI Journey

- We are embarked on a four-stage journey
  - `(Text | Image)` → `(Text | Image)` in a chat window
  - Then within the confines of an authoring tool
  - We made the LLM its own separate entity
  - We intend to interact with it as a colleague
- What's changing is:
  - The degree of autonomy we grant
  - The ability to access external systems




ChatBot
Virtual Employee
LLM
Copilot
Agentic

Everyone Gets a Copilot!

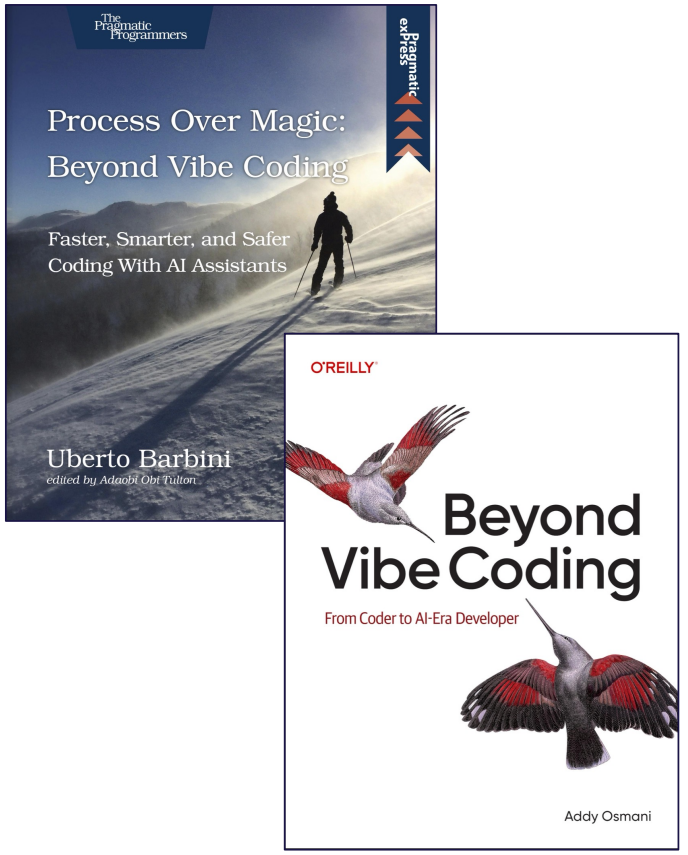


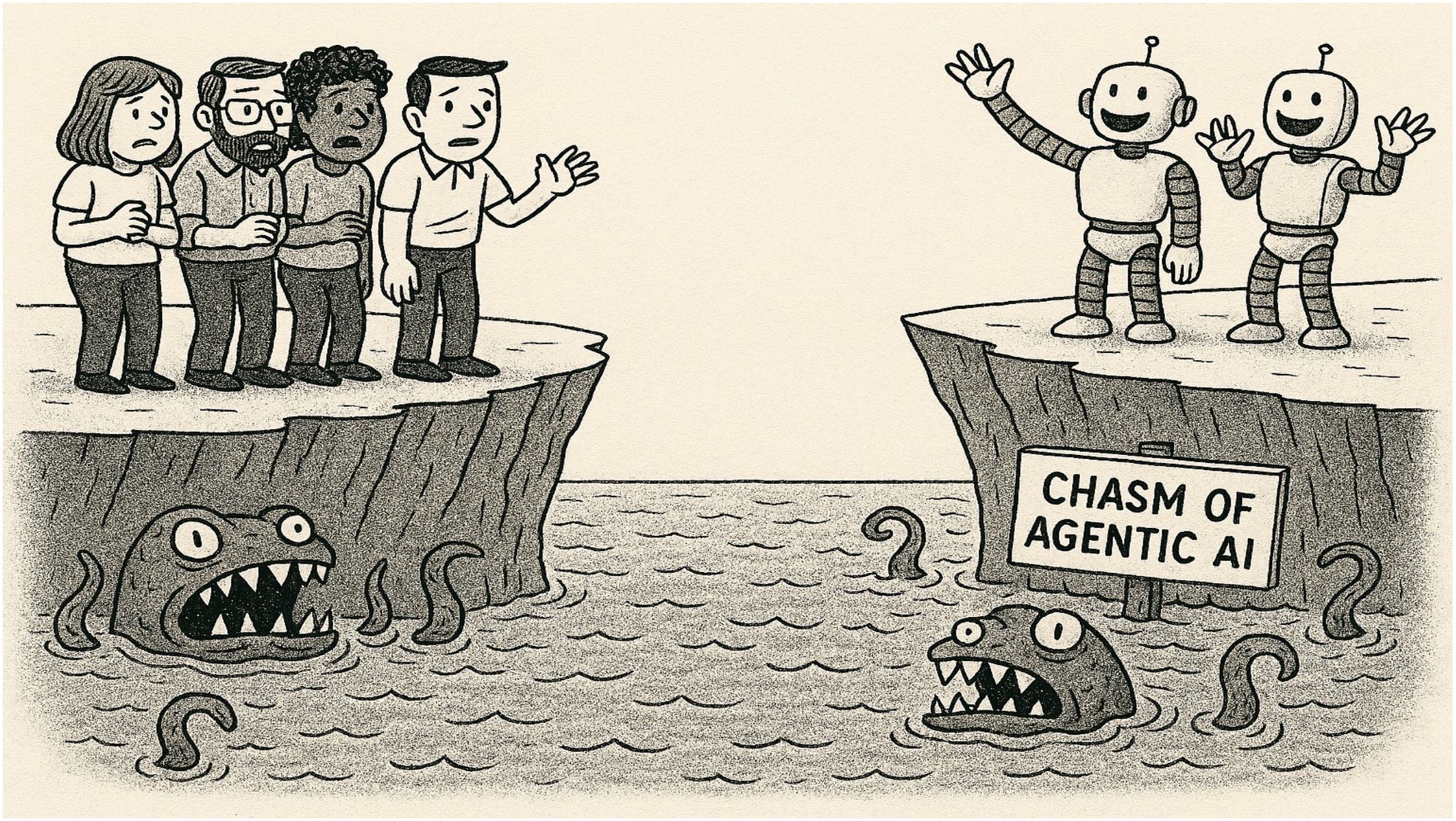
- Microsoft has 80 Copilot products
  - As of 6th April 2026
- This makes generalization hard
  - But we can agree they are legion

Source:
https://teybannerman.com/strategy/2026/03/31/how-many-microsoft-copilot-are-there.html

VibeCoding++

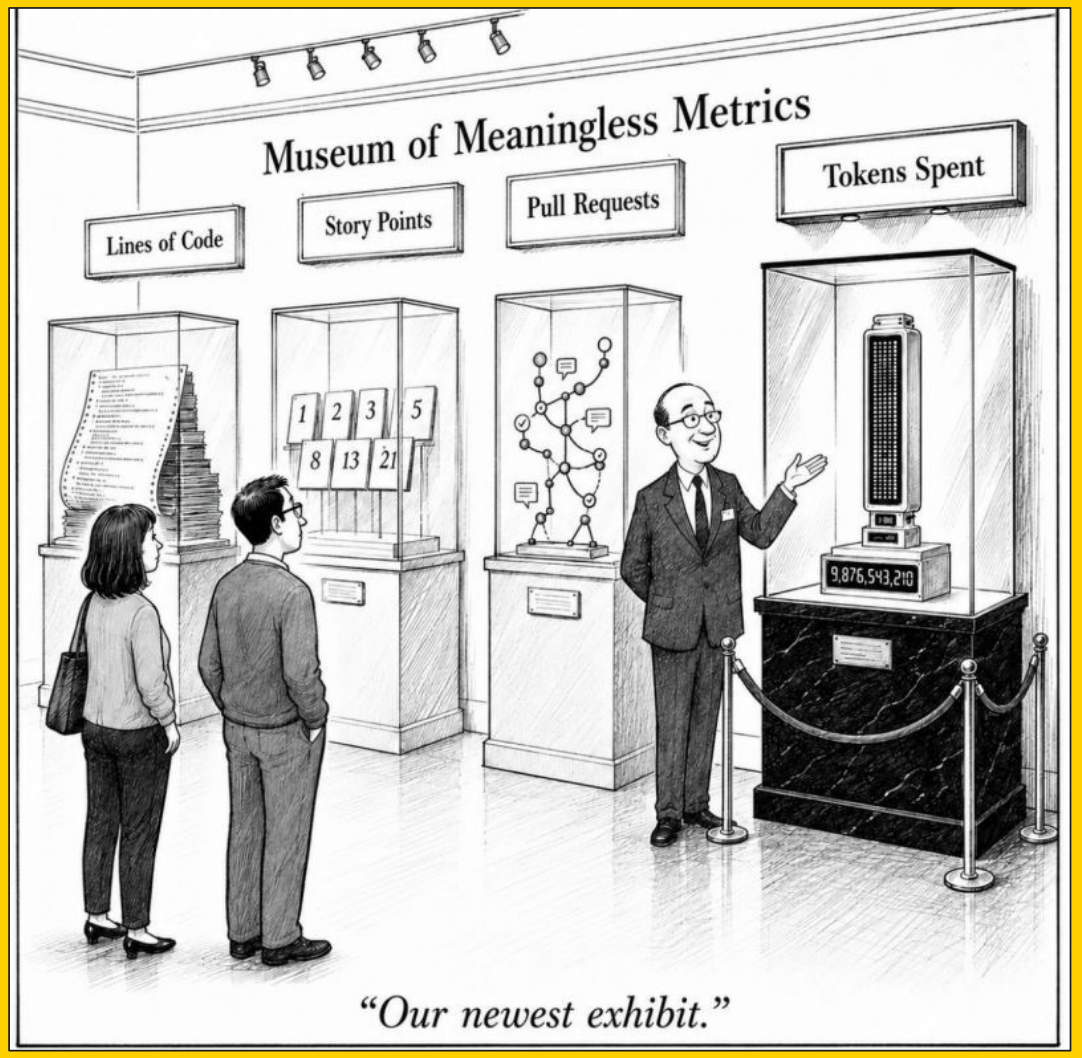
- The term Vibe Coding is barely one year old
  - Its persistence has already outlived its usefulness
- Spec Driven Development is more plausible
  - We constrain our agent via rigorous specifications
  - These cover coding standards, use of version control, testing, observability etc.
- How then do we manage fleets of Agents?
  - This is the chasm that we need to bridge before Agentic AI goes primetime




CHASM OF
AGENTIC AI

Outdated, Inaccurate or Misleading (IMO)

- The Calculator, Slide Rule, and Abstraction metaphors
- Vibe Coding and Token Maxxing 🤞🤞🤞
- Prompt Engineering as a profession
- ‘Everyone will be ten times more effective’
- ‘Knowledge work roles will be deprecated’
- ‘This is the worst model you will ever use’
- ‘You just push a button to build the software’
- ‘GenAI will lead to AGI and Quantum Chips’

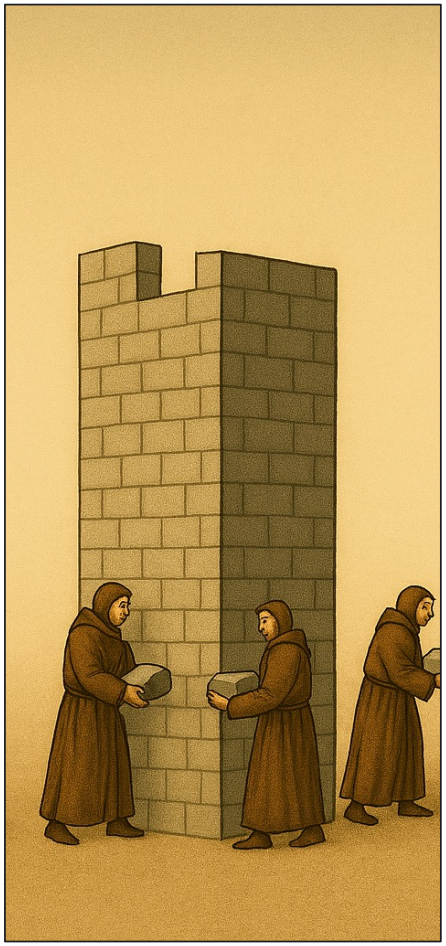

Museum of Meaningless Metrics
Lines of Code
Story Points
Pull Requests
Tokens Spent
1
2
3
5
8
13
21
9,876,543,210
"Our newest exhibit."

Emerging Skills And Risks

Skill Name	Description
AIOps / LLMOps	Controlling which models are used, how they are used, and at what cost
Agentic Orchestration	Managing teams of Agents, potentially organized in hierarchies
AI-Native Engineering	Developing software using Agentic AI as the preferred and primary mechanism
Sovereign AI	The ability to leverage AI whilst retaining control of data, infrastructure, algorithms, etc.

Risk Name	Description
Cognitive Overload	Mental burnout experienced from trying to monitor and direct teams of Agents
Cognitive Offloading (aka. Decay or Decline)	Loss of expertise from using GenAI for tasks normally done manually
Reverse Centaur	A human being whose job is to enable a machine, at the machines pace
Automation Blindness	Learned helplessness to detect issues when humans are supervising agents
Accountability Sink	A person or system that prevents change by diverting blame from the root cause

Part 2 – Does GenAI Help?



Does GenAI Help?

- Low hanging fruit for students and educators
- Using GenAI to correct the classic traps

Low Hanging Fruit for Students

- Self study at your own pace
- Fill in the ’20-year gap’ in material
- Recommend books, papers, and talks
- Ask questions as many times as you need to
- Ask questions as many ways as you need to
- Create example programs to illustrate language features
- Create complete applications to illustrate frameworks
- Explain one language / framework in terms of another

Low Hanging Fruit for Educators

- Quickly create courseware and presentations
  - With compelling visualizations of ideas and concepts
- Provide realistic sample data for example systems
  - Such as names, addresses, and postcodes for a database
- Generate quizzes and sample programs for exercises
- Break writers block and critique existing materials
  - Commonly referred to as the ‘Thought Partner’ approach
- Translate content between learning modalities
  - Generate images / audio from text and vice versa

What About The Traps?

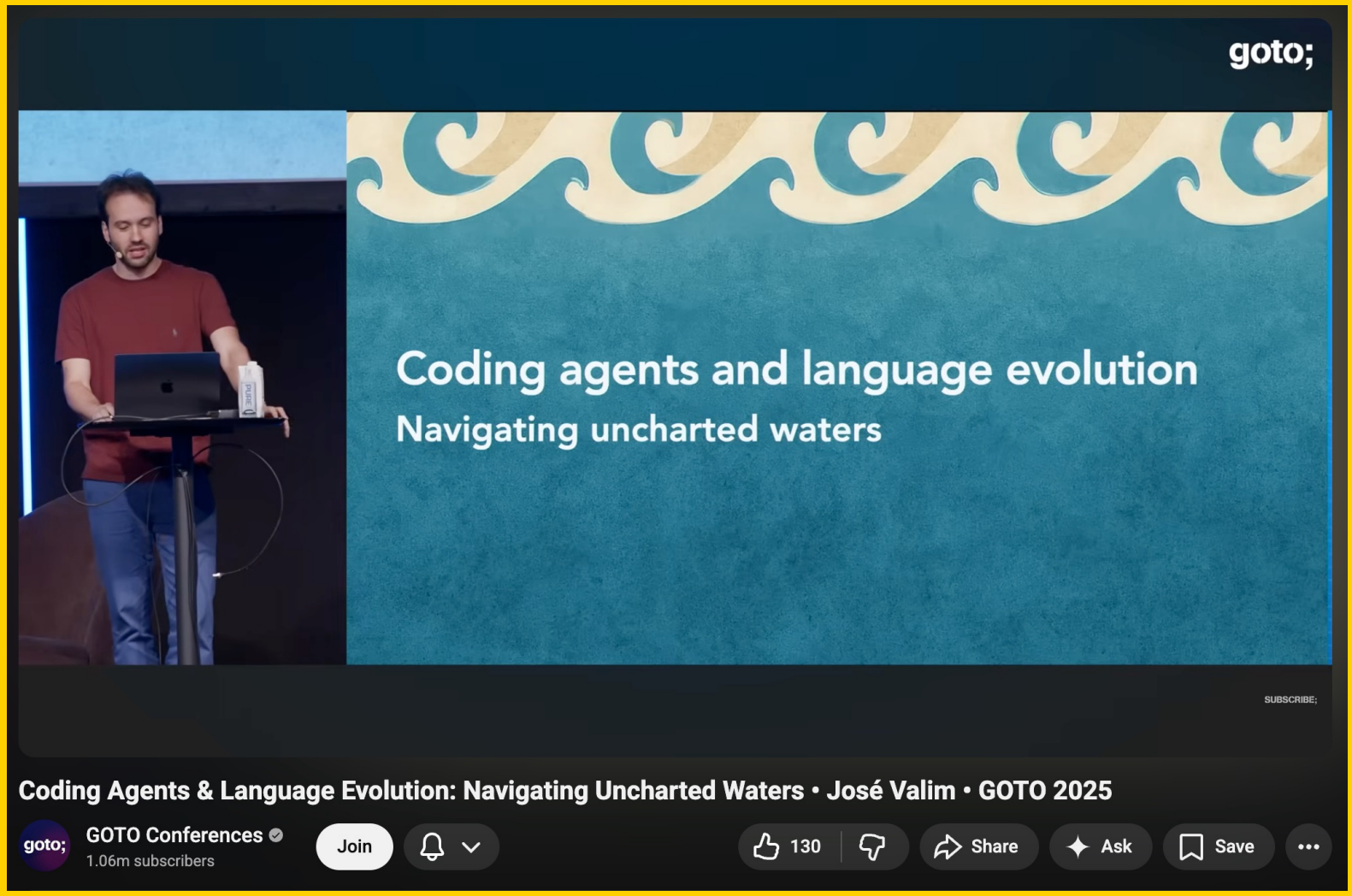
- Falling between two stools
- Not keeping pace with modernity
- Using obscure languages and tools
- Continuing to support obsolete platforms
- Omitting project work and version control
- Not communicating the realities of job roles

Falling Between Two Stools

- Even pre-GenAI we have solutions
  - There are many <u>many</u> resources to help us bridge the gap with industry
  - Many institutions are working hard to close the gap and ‘keep it real’
- We still need to find and exploit them
  - GenAI can make this process easier
- But it requires time and effort
  - That is a political and economic issue

Not Updating Materials

- GenAI is a massive help here
- It will enable educators to:
  - Create new courses from scratch as new technologies arrive
  - Update existing courses as languages and frameworks evolve
  - Critique an existing syllabus and prioritize the required work
- There are indirect benefits as well
  - Creators of new technologies will need to make them ‘GenAI friendly’
  - They will need to imbed copious documentation and exemplars



https://www.youtube.com/watch?v=VZcDxkFj_9E

Using Obscure Languages and Tools

- GenAI can lower the bar for multi-language courses
  - AI can convert examples between languages
  - Students can use AI to see through syntax
- This opens the door to some interesting exercises
  - “Here is a partially FP application in Java/C# convert it to Haskell”

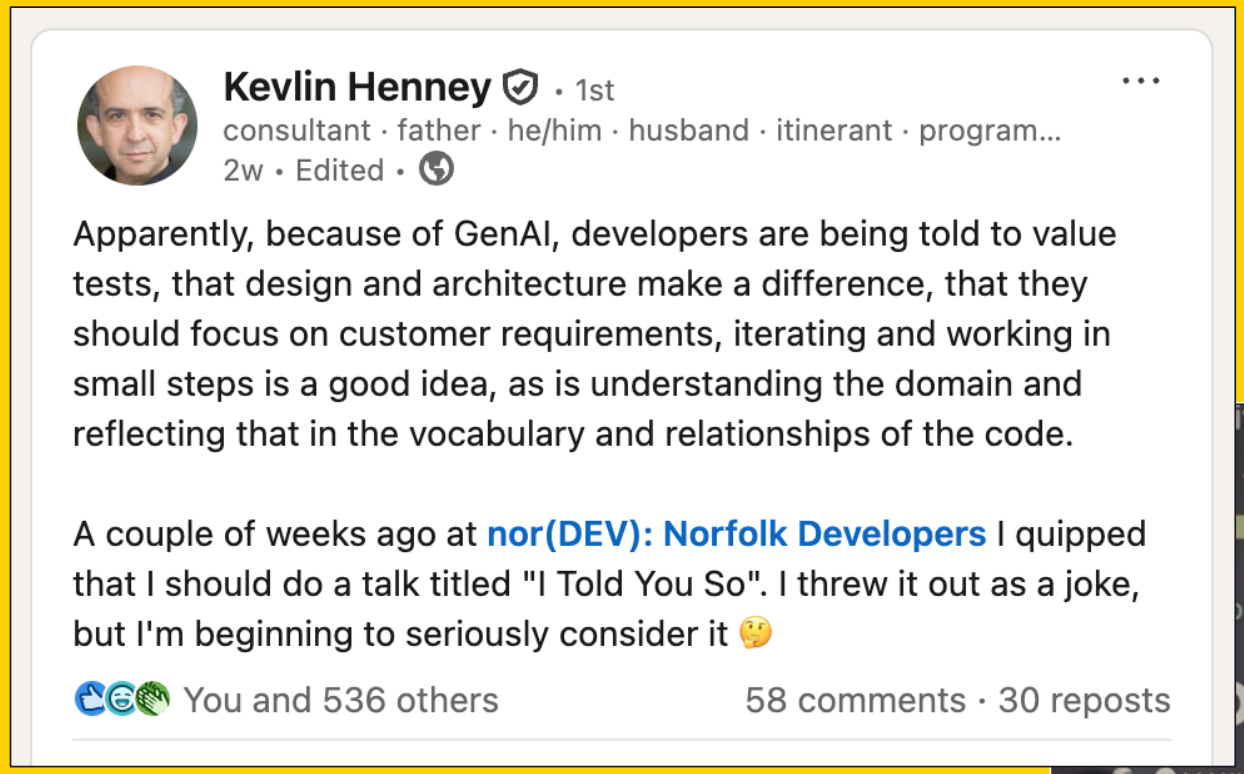
https://www.youtube.com/watch?v=aSXaxOdVtAQ

Continuing to Support Obsolete Platforms

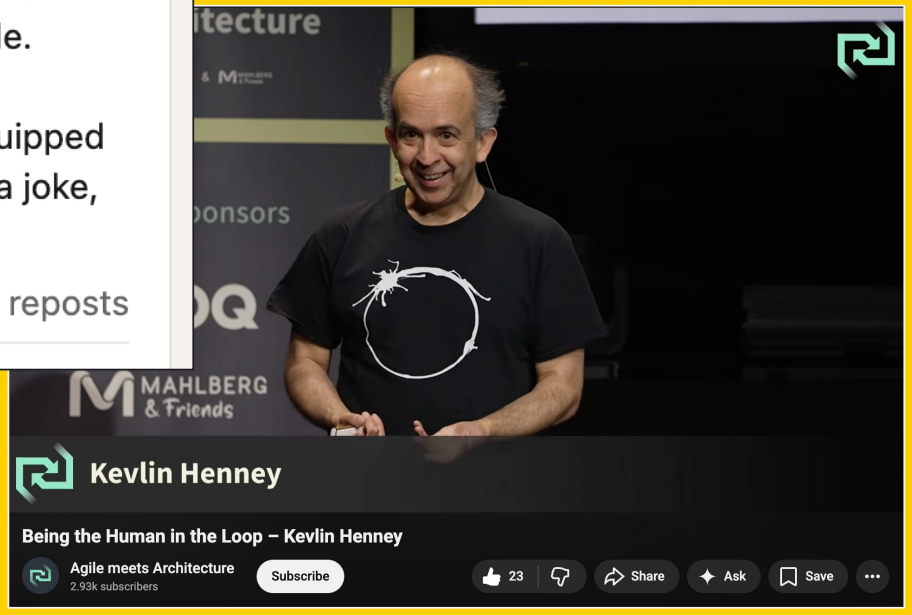
- The cadence your institution adopts is a governance issue
  - But GenAI provides a powerful argument to shorten the cycles
- A cadence measured in years has always been unrealistic
  - Unless you drop all ambitions to track software engineering
- You can now make the argument for iterative refactoring
  - Which has been the case for decades outside academia
  - The good news is it’s less painful in the long run

Omitting Project Work and Version Control

- GenAI is a massive enabler of project work
  - It really helps students build and sustain momentum
  - They can find their own way around potential blockers
- At Liberty IT we find it's invaluable during hackathons
  - Mentors don't lose time on syntax and configuration issues
- Working with Agents is a great way to embrace Git
  - Using branches and small commits is an absolute necessity
  - It turns out Agentic best practices are Agile best practices ☺



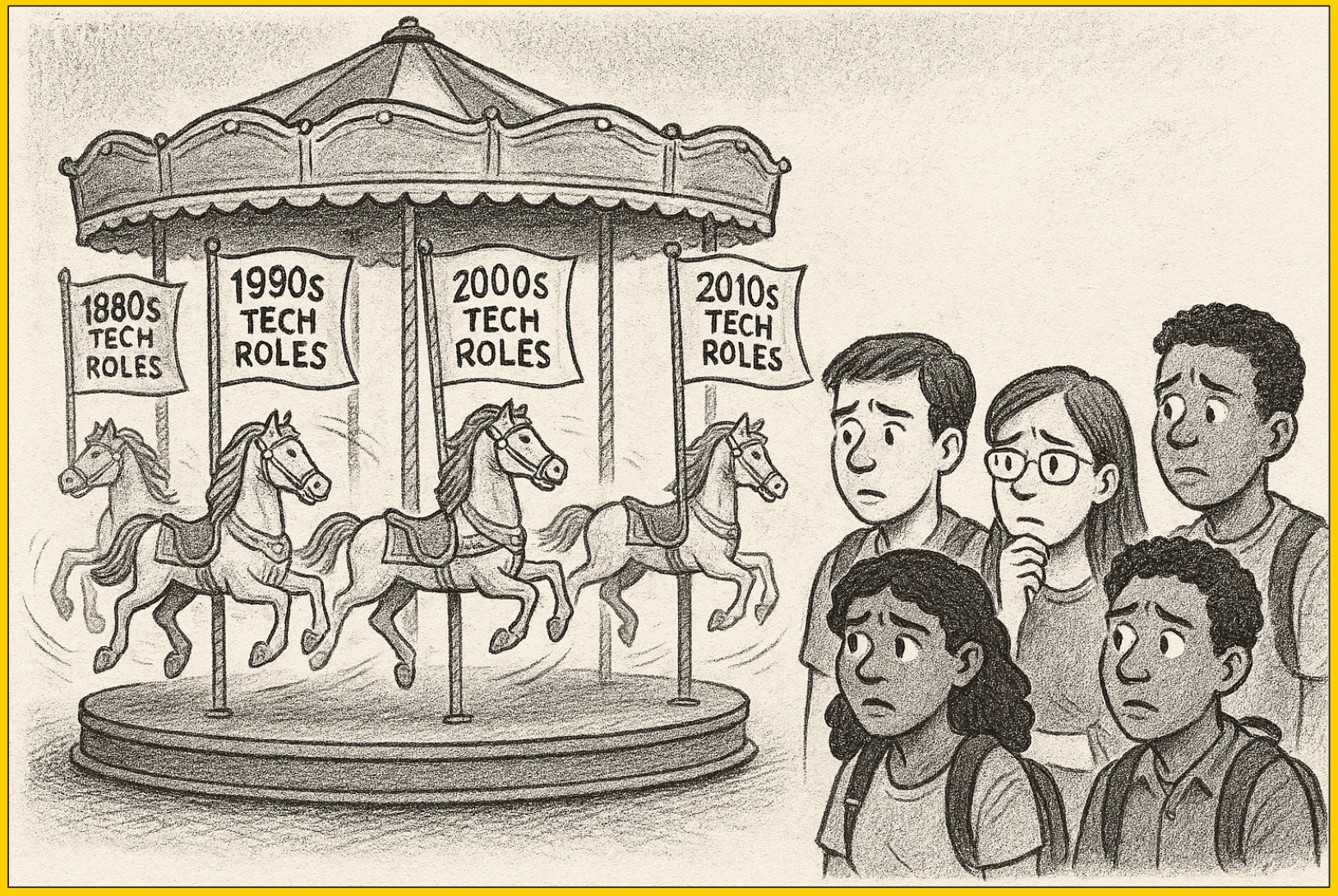
Source: https://www.linkedin.com/posts/kevlin_apparently-because-of-genai-developers-activity-7439626645180665858-gsrE



Source https://www.youtube.com/watch?v=vpYJMr1pJRY

Not Communicating the Realities of Job Roles

- GenAI offers an opportunity here to reset the clock
  - Or as I like to think of it ‘step back onto the merry-go-round’
- Every so often the roles within SoftEng rebalance themselves
  - Old roles like Systems Analyst and Web Designer are deprecated
  - New roles like Product Owner and Site Reliability Engineer emerge
- GenAI is forecast to cause a sea change in responsibilities
  - So now is a great time to join in with the conversation!


1880s TECH ROLES
1990s TECH ROLES
2000s TECH ROLES
2010s TECH ROLES

Conclusions

Summing Up

- The CompSci vs SoftEng problem is an enduring one
  - Because it is Conways Law at the level of human society
- IMO GenAI is the best tool to help with this since the pencil
  - If you ignore the existential threat to our profession ☺
- Using GenAI we can:
  - Facilitate curiosity and self-learning as never before
  - Find new ways to make the timeless lessons engaging
  - Adapt to new platforms, frameworks, and languages
  - Create graduates who share our curiosity and passion!

Stay connected

Follow us on our social channels for a first look at our job opportunities and a glimpse of life at Liberty IT.

Liberty™ Information Technology


Liberty™
Information
Technology