University of Sheffield

COM1001 SPRING SEMESTER
Professor Phil McMinn
p.mcminn@sheffield.ac.uk

Is Software Engineering Dead?

Today's attendance code: BB-RT-AV

GitHub CoPilot

An AI-Powered “Code Assistant”

Uses **Large Language Models** to suggest or generate code, based on natural language prompts and current code context

- Developers write descriptions like *“implement quick sort in Python”*

First launched in June 2021

Powered by generative models like OpenAI’s Codex that were **pre-trained on a large corpora of natural language and publicly available source code** (including public GitHub repositories)

More Recently: LLMs beyond Autocomplete

- LLMs have helped architects co-design new languages and large systems
- Systems like AplhaEvolve have adapted and refined new algorithms autonomously
- **Only last week:** 16 LLM agents wrote a 100k+ line C compiler capable of compiling Linux in ~2 weeks

Building a C Compiler with a Team of Parallel Claudes

https://www.anthropic.com/engineering/building-c-compiler

16 autonomous LLM agents powered by Anthropic's Claude Opus 4.6 wrote a C compiler from scratch in Rust with **minimal human direction**.

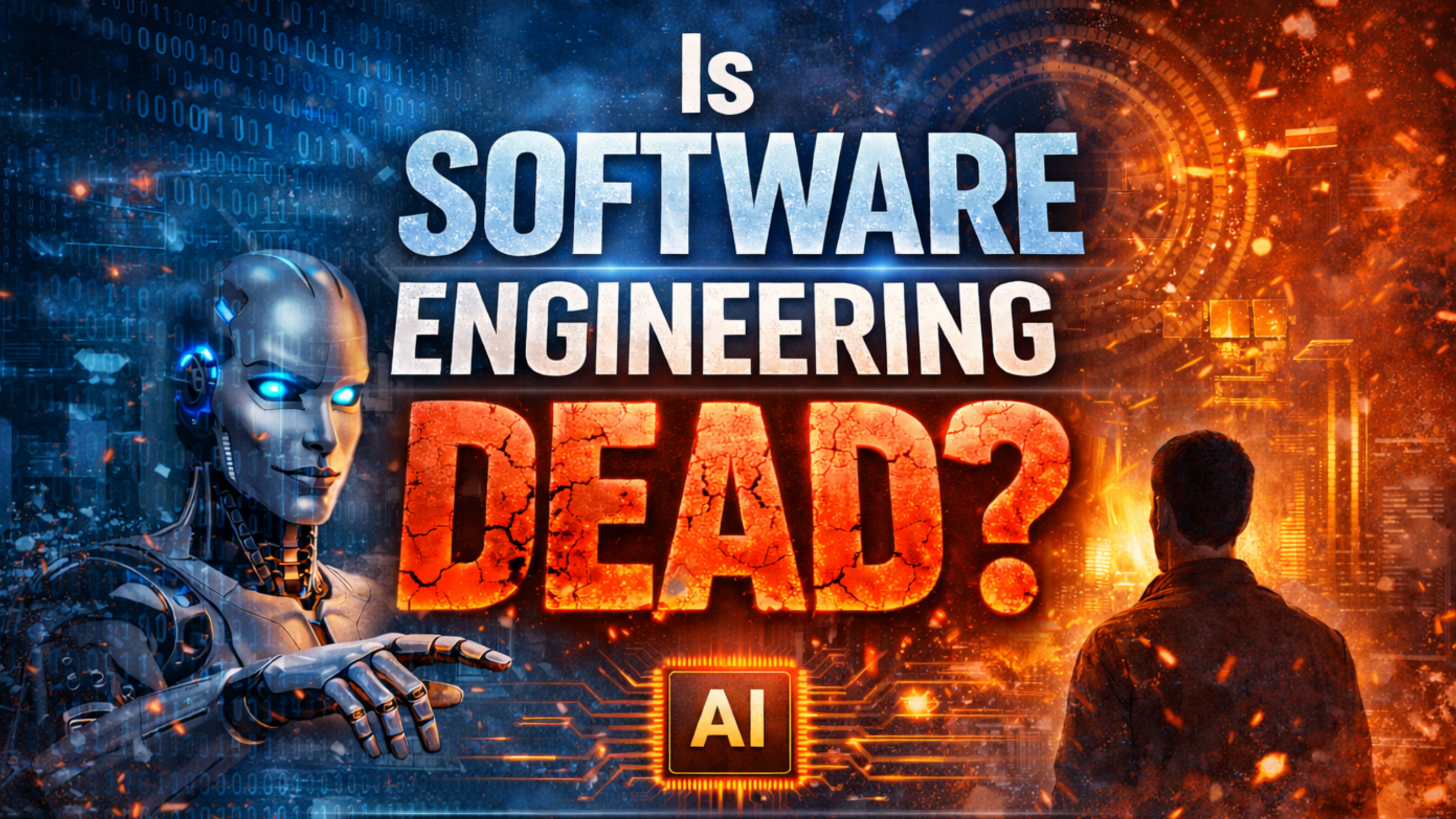
Resulted in a 100k line compile capable of handling major real world software, including the Linux kernel.

Cost around $20k in API calls and two weeks of compilation

Why this matters: writing a compiler is considered one of the hardest tasks in software engineering.

All of this raises new questions about…

- Are programming skills actually going to be needed anymore?
- Can we trust code generated by AI?
- What does it mean to be a software engineer?


Is
SOFTWARE
ENGINEERING
DEAD?
AI

So…

Is software engineering dead?

What do you think?

So…

Is software engineering dead?

Yes!

So…

Is software engineering dead?

Yes!

… as we know it

Things to Remember…

Firstly, **software engineering is more than just programming:**

- There's communicating with stakeholders, ensuring software is fit for purpose (verification and testing), debugging, maintenance.

Secondly, AI *is* changing software development. **But actually, if we look closely "AI" has been doing that since the inception of the field itself.**

This is a lecture about AI in Software Engineering

The Past, the Present, and the Future

I want to talk about how “AI” has been in software engineering for longer than you might think.

I want to spend a bit of time on **where we are now**.

Importantly, I want to talk about **where we might be going** and **what that means for you** now, and **how you engage with COM1001**.

AI started in software engineering with tools that helped developers.

Many developers were using AI before they called it AI:

> Autocomplete, code search, automated bug detection, fixing and automated test generation were all techniques *before* LLMs

2022/3: Code generation using LLMs gained visibility through tools like **ChatGPT**. This brought a big shift in attention onto AI!

Currently there are lot of layoffs going on in the tech sector. **Is this because of AI or is something else happening?** I'll come back to this later…

AI in SE: A Brief Timeline

1950-1980s: Foundations

1950 – Turing proposes the Turing test

Early AI focussed on symbolic reasoning and rule-based systems

1990s: Smarter Tools

Tools for program analysis – static and dynamic analysis

Limited automatic bug detection, test case generation

2000s: Search and IDE uptake

Search techniques (e.g. Genetic Algorithms) used to generate test cases

Static analysis in IDEs for automated refactoring

2010s: Machine Learning

- ML used to predict bugs, suggest refactoring.
- IDEs become more intelligent

AI in SE: A Brief Timeline

2016-2019: Deep Learning

- Neural models trained on large open source codebases
- Code completion improves significantly

2020-2022: Large Language Models for Code

- Models trained on natural manage and code
- Tools can generate functions, explain code, translate between languages

2023-present: AI as a programming partner

- AI integrated into IDEs, relied on for code generation
- Software agents that can work independently and report back larger solutions

Some Theories About The Future

AI completely automates programming. Programmers are not needed any more. Programmers get made redundant.

AI automates most of programming. Programming is now mostly supervising the AI – reviewing its code and writing tests

AI means we only need expert programmers, junior devs are replaced with AI.

AI lowers the cost of programming. Building software is cheaper. So actually more programmers are demanded (the so-called “Jevons” economic paradox).

AI neither increases or decreases the cost of software engineering. Expert devs don’t need it. For others, the costs of dealing with and managing “unsafe” AI-generated code outweigh the speed gains.

The Way I See It* - Part 1

Many people think this is an instance of the horse 🐎 to the car 🚗.

But, we have a tendency as humans, to take things now and then extrapolate them.

- Flight ✈️ to Deep Space 🚀 🪐
- Is this going to be an airplane or a horse?

We don’t know what the future holds – nobody does:

- Nobody knows what’s even going to happen in the next 6 months.
- Think back to COVID … nobody really knew what was going to happen, remember Boris and the “experts”

Let’s just relax a little bit? And try to enjoy watching this technology grow.

** This is all personal opinion, not academic fact!*

The Way I See It - Part 2

- **AI is unlikely to completely automate programming**, although for some roles, the future of development is going to be largely spent reviewing AI-generated code.

- Given its training set, **AI is great at automating a lot of general coding tasks, but poor at edge cases and very domain-specific tasks.** LLMs generate code by pattern completion, not by extrapolating or by good planning.

- **The fact AI probably won't get to 100% means we will still need humans in the loop**.

  - This is why self-driving cars are not seeing mass deployment, nor are other "at-risk" jobs being made redundant (the demise of radiography was predicted as early as 2016, but they're still very much around).

  - And … ultimately humans need other humans to blame 😆

  - **History is dull but reassuring:** mechanisation didn't end farming. Computers didn't end office work. The internet did not end retail.

The Way I See It - Part 3

Developers will go from predominantly writing code to reviewing code.

There will be new “programming languages” that are very close to natural language, or what we now call pseudo-code.

“Coding” will be about thinking clearly and reviewing well.

The edge won’t be **speed** anymore, it will be **judgement.**

I think we probably will need fewer programmers in the future, but those programmers will be 🔥.

There will be less room for bad programmers.

Because if I can just get an LLM to do it, can’t anyone do that ?

So What Does This Mean for You?

In the short to medium term – although development may be largely centred around prompting AIs to generate code, we still need to understand it to review it and test it.

To be able to review it, we still need to be able to write code ourselves, if needed!

But how do we learn to write code in this new age?

Same as before! And importantly, without LLMs:

- To learn the thing, you have to do the thing
- Nobody got good at guitar by just watching people play guitar.
- Nobody got a six-pack from buying a gym membership. They had to go to and use the gym.

Ruby and Humans

The great thing about Ruby is that it is very close to English already.

It’s one of the most readable, human-friendly languages out there.

> *“I want to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the purpose of Ruby. I emphasize the word ‘every.’”* — Matz

Yukihiro Matsumo
(“Matz”) – creator of Ruby

Ruby on Rails underpins Major Internet Businesses

- Shopify is a platform built on Rails that enables millions of merchants to sell online, turning over $14.6 billion on Black Friday to Cyber Monday in 2025 (up 27% on the prior year)
- GitHub is famously built on Rails.
- Others: Airbnb, Fiverr, Etsy, Groupon…

Sinatra is a great way in to learning Ruby, and later, Rails.

- Widely used commercially too, but less visibly! Internal services, APIs and micro services.

David Heinemeier Hansson
(“DHH”) – creator of Ruby On Rails

How AI Impacts Skill Formation

Judy Hanwen Shen* Alex Tamkin†

February 3, 2026

Abstract

AI assistance produces significant productivity gains across professional domains, particularly for novice workers. Yet how this assistance affects the development of skills required to effectively supervise AI remains unclear. Novice workers who rely heavily on AI to complete unfamiliar tasks may compromise their own skill acquisition in the process. We conduct randomized experiments to study how developers gained mastery of a new asynchronous programming library with and without the assistance of AI. We find that AI use impairs conceptual understanding, code reading, and debugging abilities, without delivering significant efficiency gains on average. Participants who fully delegated coding tasks showed some productivity improvements, but at the cost of learning the library. We identify six distinct AI interaction patterns, three of which involve cognitive engagement and preserve learning outcomes even when participants receive AI assistance. Our findings suggest that AI-enhanced productivity is not a shortcut to competence and AI assistance should be carefully adopted into workflows to preserve skill formation – particularly in safety-critical domains.

1 Introduction

Since the industrial revolution, skills in the labor market have continually shifted in response to the introduction of new technology; the role of workers often shifts from performing the task to supervising the task [Autor et al., 2001]. For example, the automation of factory robots has enabled humans to move from manual labor to supervision, and accounting software has enabled professionals to move from performing raw calculations to identifying better bookkeeping and tax strategies. In both scenarios, humans are responsible for the quality of the final product and are liable for any errors [Bleher and Braun, 2022]. Even as automation changes the process of completing tasks, technical knowledge to identify and fix errors remains extremely important.

As AI promises to be a catalyst for automation and productivity in a wide range of applications, from software engineering to entrepreneurship [Dell'Acqua et al., 2023, Peng et al., 2023, Cui et al., 2024, Otis et al., 2024, Brynjolfsson et al., 2025], the impacts of AI on the labor force are not yet fully understood. Although more workers rely on AI to improve their productivity, it is unclear whether the use of AI assistance in the workplace might hinder core understanding of concepts or prevent the development of skills necessary

Paper from researchers at **Anthropic** (originators of Claude)

How AI Impacts Skill Formation

Judy Hanwen Shen* Alex Tamkin†

February 3, 2026

Abstract

AI assistance produces significant productivity gains across professional domains, particularly for novice workers. Yet how this assista… required to effectively supervise AI remains unclear. Novice worker… their own skill acquisition in the… gained mastery of a new async… We find that AI use impairs co… delivering significant efficiency … some productivity improvemen… interaction patterns, three of w… when participants receive AI a… shortcut to competence and A… formation – particularly in saf…

1 Introduction

Since the industrial revolution, ski… of new technology; the role of wo… et al., 2001]. For example, the aut… to supervision, and accounting softw… to identifying better bookkeeping and tax … of the final product and are liable for any errors [Bleher and Braun, 2023]. process of completing tasks, technical knowledge to identify and fix errors remains extremely important…

As AI promises to be a catalyst for automation and productivity in a wide range of applications, from software engineering to entrepreneurship [Dell'Acqua et al., 2023, Peng et al., 2023, Cui et al., 2024, Otis et al., 2024, Brynjolfsson et al., 2025], the impacts of AI on the labor force are not yet fully understood. Although more workers rely on AI to improve their productivity, it is unclear whether the use of AI assistance in the workplace might hinder core understanding of concepts or prevent the development of skills necessary

> "We find that AI use impairs conceptual understanding, code reading, and debugging abilities, without delivering significant efficiency gains on average. Participants who fully delegated coding tasks showed some productivity improvements, but at the cost of learning the library."

Paper from researchers at **Anthropic** (originators of Claude)

What does this mean for COM1001?

Do not use LLMs to write the code for you

This is against the rules anyway!

The best use of these tools is to help you learn, not take the learning away from you.

So don't:

Ask an LLM to write code and copy and paste it!

Do:

Attempt to write it yourself

If you get stuck, by all means, *ask the LLM for advice or a second opinion*. But always take the advice with suspicion – engage your own brain 🧠

Never copy any fixes from the LLM, always type them in yourself – remember: you have to do the thing to learn the thing.

None of this means that, long term, you shouldn't use LLMs to help you write code.
But that's not this module.

What Does This Mean for Software Engineering?

Judging the output of LLM-generated code will become important.

That means the ability to **review**, **debug and test code**, and **make direct fixes**.

Testing and verification just became more important that it ever did – we will be covering this in this course.