UNIVERSITY OF CAMBRIDGE

Reliable Software and Security Engineering with Unreliable Tools

Lecture 4 - Law, Ethics, Accountability, and The Future

David G Khachaturov[1, 2]

[1]St Catharine's College, University of Cambridge
[2]Computer Laboratory, University of Cambridge

12th March 2026

Recap

- **Trust:** We rely on unreliable tools in the form of LLMs and other ML models. The attack surface has expanded to include the model, the training data, and the prompt.
- **Attacks:** AI generates “happy path” code that looks correct but fails on edge cases. We are mass-producing plausible but vulnerable code (e.g., Slopsquatting).
- **Defences:** We cannot trust the author, so we must verify the artifact. Use Type Systems, Sandboxing, and Fuzzing to make illegal states unrepresentable.

Today: Consequences

- When the defences fail, who is responsible?
- How does the profession change when “writing code” is automated?

Liability in Non-Deterministic Systems

For legal purposes: *IANAL, TINLA*.

Core Legal Problem:

- Traditional software liability typically relies on **negligence**.
  "Did the engineer fail to exercise reasonable care?"

AI Liability Gap:

- If a "Black Box" writes the code, and that code contains a vulnerability, is the human engineer negligent for not finding it?
- Is the model provider (e.g., OpenAI, Google) liable for providing a "defective product"?

Current Reality:

- End User License Agreements shield vendors ("*AS IS*"). This shield is cracking under consumer protection laws.

Statutory Regulation: “Black Box” Problem

Civil Liability (i.e., *getting sued*) is only half the risk. Regulatory Compliance (i.e., *breaking the law*) is the other.

Transparency Paradox:

- GDPR Article 22 / EU AI Act “*Right to Explanation*”
- If a system makes a significant decision (credit, hiring, safety), you must be able to explain *how* it reached that decision

Conflict:

- Deep Learning models are often opaque
- Formal Verification (Lecture 3) is not just good engineering; it is becoming a legal requirement for “High Risk” AI systems

If you can’t explain it, you can’t deploy it.

Moffatt v. Air Canada (2024)

Facts:

1. Mr. Moffatt asks Air Canada's AI chatbot about bereavement fares.
2. The Chatbot hallucinates a policy: "*You can retroactively apply for a refund of the difference between regular and bereavement fares within 90 days.*"
3. The actual policy (on a static webpage): no retrospective refunds.
4. Moffatt buys the ticket, applies for a refund, and is rejected.

Defence: Air Canada argued the chatbot was a "*separate legal entity that is responsible for its own actions*".

Ruling: The British Columbia Civil Resolution Tribunal rejected this defence. **Air Canada was held liable for negligent misrepresentation.**

> *"It should be obvious to Air Canada that it is responsible for all the information on its website, it makes no difference whether the information comes from a static page or a chatbot."*

“Human in the Loop” Legal Fiction

Corporations often use “Human in the Loop” (HITL) as a liability shield.

“AI didn’t make the decision; the assigned human clicked approve”

Moral Crumple Zone

Humans are inserted into automated systems not to control them, but to absorb the liability when they fail.

— Elish, Madeleine Clare. “*Moral Crumple Zones: Cautionary Tales in Human-Robot Interaction*”. 2019

If the system is designed to induce *automation bias* (Lecture 2), the human cannot effectively supervise it.

The human becomes a *liability sponge*, absorbing the legal blame for machine errors they were statistically unlikely to catch.

Cognitive Offloading

Cognitive Offloading

Use of physical or digital resources to reduce the cognitive demands of a task.

Short-term:

- Massive productivity boost. You don't need to remember syntax, boilerplate, or API references.

Long-term:

- Erosion of expertise
- e.g., "*GPS Effect*"; drivers who rely solely on GPS suffer from hippocampal atrophy and lose the ability to navigate spatially
- What happens to the "*Coding Brain*" when we stop writing code?

Junior Developer Paradox

Traditional Model:

Junior writes simple code $\rightarrow$ Senior reviews it $\rightarrow$ Junior learns and improves

AI Era:

- AI writes simple code instantly
- Junior is promoted to "AI Supervisor"

Conundrum:

- How do you supervise code you don't understand?
- How do you become a Senior Engineer if you never struggled through the "grunt work" that builds mental models?

We risk creating engineers who can *Prompt*, but cannot *Debug*.

Ironies of Automation

> *"The more advanced a control system is, so the more crucial may be the contribution of the human operator."*
> *— Lisanne Bainbridge, 1983*

When the AI works 99% of the time:

1. The human operator zones out (*vigilance decrement*)
2. The human loses their mental model of the system
3. When the AI fails (the 1%), it is usually a catastrophic edge case
4. The disengaged human is expected to fix the hardest problem instantly

Result: Brittle Failure (Lecture 1)

Safety and Security

We often conflate these terms, but they are distinct disciplines:

- Security Engineering (Anderson); focuses on *malice and the intelligent adversary*; goal is to *prevent intentional subversion*
- System Safety (Leveson); focuses on *complexity and accidents*; goal is to *prevent unintended loss of life/property*

These term *converge* in two contexts – *IoT* and *AI*:

- *IoT*: Anderson, Clayton, and Leverett (2015) noted that with software everywhere safety and security get entangled (and in many EU languages, they are the same word!)
- AI hallucination is a **Safety** failure (unintended).
- However, an attacker can *provoke* it (via e.g., prompt injection), making it a **Security** failure

Engineering a Safer World

It is worth bearing in mind Nancy Leveson's Systems-Theoretic Accident Model and Processes (**STAMP**)

Key insight:

- Safety is an **emergent property** of the system, not a component property.
- You cannot make a system safe just by having "safe" components.
- Reliable components can interact to cause accidents.

Relevance:

- We cannot fully verify the "AI Component"
- Therefore, we must secure the **System Architecture**

e.g., the database (*System*) must reject the AI's SQL Injection (*Component Failure*).

Ethics and Responsibility

> *"Cryptography [& Engineering] rearranges power: it configures who can do what, from what. This makes cryptography* ***an inherently political tool****"*
> *— Rogaway, Phillip. The Moral Character of Cryptographic Work. 2015*

Neutral Tool Fallacy:

"I just build the hammer; I don't decide how it's used."

Rogaway's Argument:

- Technology embodies the values of its creators.
- If you build a system that relies on opaque AI to deny loans, filter CVs, or target weapons, you are *making a political choice to prioritise efficiency over justice*.

Professional Standards

Ethics, beyond a philosophy, is a *professional requirement*.

ACM Code of Ethics (2.5):

> *"Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks."*

BCS Code of Conduct (2.b):

> *"You shall not claim any level of competence that you do not possess."*

- If you deploy a "black box" you cannot explain, you are failing your evaluation duties.
- If you use AI to write code you do not understand, you are failing your competence duties.

Dual-Use Technologies

The tools we have studied in this course are fundamentally *dual-use*.

Automated Vulnerability Discovery:

- *Defender:* Finds bugs to patch them
- *Attacker:* Finds bugs to exploit them

Generative AI Coding:

- *Defender:* Writes secure tests and boilerplate
- *Attacker:* Generates polymorphic malware and phishing campaigns

The only defence against an attacker is **Resilience**. We must build systems that assume the attacker has the same tools we do.

Future of the Profession

Is the “Programmer” dead?

> *“NVIDIA CEO says kids shouldn’t learn to code.”*

Bear in mind that NVIDIA CEO also sells the shovels.

Counter-argument:

- *Coding* (Syntax) is becoming a commodity
- *Engineering* (Semantics, Verification, System Design) is becoming a luxury

Job shifts from **Translation** (Human $\rightarrow$ Code) to **Verification** (Code $\rightarrow$ Proof).

1. *Specification:* Defining what the system should do
2. *Constraints:* Enforcing safety boundaries
3. *Accountability:* Taking responsibility for the failure

Economic Drivers: The Insurance Market

> *"**Cyber-insurance** will eventually drive security standards."*
> *— paraphrasing Ross Anderson's words from Security Engineering*

This view has been mirrored by Professor John Hasnas in his recent Dec 2025 lecture "*Common law: a better foundation for Liberalism*".

As liability clarifies (Air Canada, EU AI Act), insurance premiums for "unverified AI code" will skyrocket.

Insurers will likely demand:

1. Software Bill of Materials (SBOM)
2. Proof of Verification (Formal Methods / Fuzzing logs)
3. "Human in the Loop" audit trails

Security Engineering becomes a function of **Risk Management**.

Summary

1. **Skepticism:** Treat AI output as “untrusted input” from a potentially malicious user.
2. **Constraints:** Use strong types, sandboxes, and formal verification to make illegal states unrepresentable.
3. **Accountability:** You are the moral agent. The machine is just a probabilistic parrot.
4. **Expertise:** Don’t stop learning the fundamentals. You need to know *why* the code works, because the AI only knows *that* it looks right.

We, the Feeble Minded

An extract from Isaac Asimov's novella "*Profession*" (1957).

Context: In a future where everyone learns professions instantly via "taping" (direct brain upload), the protagonist is rejected and sent to an institute for the "feeble-minded" because he cannot be taped. He eventually discovers the truth:

> *"…who invents the new instrument models that require new-model technicians? … he couldn't have been tape-Educated or how could he have made the advance? … who makes Educational tapes? Special tape-making technicians? Then who makes the tapes to train them? … Somewhere there has to be an end. Somewhere there must be men and women with* ***capacity for original thought****."*

Recommended Reading

1. Chapters 2, 3, and 25 of **Anderson, R.** *Security Engineering* (3rd Ed.)
2. **Rogaway, P.** *The Moral Character of Cryptographic Work*. Essay. 2015
3. Chapter 9 of **Leveson, N.** *Engineering a Safer World*. MIT Press
4. **Elish, M.C.** "*Moral Crumple Zones: Cautionary Tales in Human-Robot Interaction*". 2019
5. **Asimov, I.** *Profession*. Astounding Science Fiction. (1957)

Questions?

Course page with feedback form and recommended reading

David G Khachaturov

dgk27@cam.ac.uk

Thank you for your attention