

The Actuary Who Only Prompts Is Not an Actuary

AI will revolutionise our work. It is touted as the fifth industrial revolution. I have no doubt it will transform how we work. It already has.

You can see it most easily in writing. AI writing is bland, overly polished, and overly dramatic. It is filled with performed gravity, empty conclusions, and false inevitability. It sounds good. Really good.

But is it?

Epistemic Vigilance

If Swiss cheese has holes, then more Swiss cheese has more holes. More holes means less cheese. Therefore, more cheese means less cheese.

This is a whole new kind of shrinkflation.

The argument is obviously false, but it takes a moment to find the problem. "More holes means less cheese" assumes that we are comparing two pieces of the same size. "More Swiss cheese has more holes" changes the size of the piece. The logic moves between those two meanings without telling us.

The joke works because each sentence sounds reasonable on its own. The conclusion follows neatly, provided we do not inspect the assumptions too closely.

This is epistemic vigilance: taking the time to ask why an answer should be believed. What assumptions does it rely on? What information produced it? What has been left out? A convincing answer is not necessarily a correct one.

On the surface, things can appear convincing, poised, polished, and perfect. Possibly even calm and ready. Spend time with them, pick them apart, and you may find that they are bland, empty, performative, and hollow. Surely not mom's spaghetti.

That thought is worth sitting with. AI is quietly creating a shift in how we do things. The problem is not that AI cannot write. It is that AI can write. And that

carries real consequences. The question is no longer when AI will replace us, but why on earth we think we are that replaceable.

Pause and think about how you felt reading that last paragraph. Then, in contrast, ask whether an AI would reference Eminem's sweater in a serious essay about AI and the future of the actuarial profession.

If the writing is bad but looks good, what about the rest?

AI writing is the easy example. Consider the same model writing code, automating processes, and making decisions. Should we trust it? How should we use it? More importantly, how will we know when it is wrong?

I think AI will be what we make of it. It is powerful and useful. I am very pro-AI. I use it daily. I am also a sceptic. We risk losing a great deal if we use it without understanding what has changed.

The change is the negligible cost of producing a plausible answer. Problems can now appear solved almost instantly, with little effort or thought from us. AI lets us produce the artefacts of thought without doing all the thinking those artefacts imply. The report exists. The code runs. The model produces an answer. Their existence usually suggests that someone has worked through the problem. That relationship is breaking.

Yes, AI is a tool. It is also a tool that can automate parts of thought. We cannot afford to outsource the whole process. Thought got us here. Thought defined the problems, challenged the assumptions, and decided which answers were worth acting on.

No Silver Bullet

Fred Brooks gave us a useful way to separate the work we should gladly automate from the work we must preserve. In *No Silver Bullet*, he distinguished accidental complexity from essential complexity in software.¹

Accidental complexity comes from the tools and processes used to implement a solution. It is boilerplate code, awkward syntax, manual copying, and the effort

¹Frederick P. Brooks, Jr., "No Silver Bullet: Essence and Accidents of Software Engineering", *Computer* 20, no. 4 (1987): 10–19. <https://doi.org/10.1109/MC.1987.1663532>.

needed to make one system speak to another. There is no virtue in spending an afternoon fixing brackets or copying the same data between workbooks.

Essential complexity comes from the problem itself. What must the system do? What does the data represent? Which constraints matter? What happens when an assumption fails? Should the process exist at all?

Large language models are excellent at reducing accidental complexity. They can generate boilerplate code, transform data, draft documentation, and explain unfamiliar libraries. They make technical work more accessible to people who would otherwise struggle to begin. I use them for exactly these things.

They do not remove the essential complexity. A generated script does not establish that a data issue has been correctly diagnosed, that the data reflects how the business operates, or that the process should be automated. A fluent explanation of a claims movement may offer useful hypotheses. Evidence is still needed to establish the cause. An automated decision can process the information it receives, but it cannot account for information nobody thought to provide.

Dijkstra made a related point in his essay on natural-language programming.² Natural language is useful for describing an intention, but it is not a specification. A prompt can state assumptions, boundaries, failure conditions, and tests. Putting these matters into prose does not show that they are complete, internally consistent, or valid.

Some may argue that agents have moved beyond static prompts. Modern frontier models can act autonomously, chain tools, and reason through multi-step objectives. The evidence can look formidable: OpenAI reported that an evaluation model escaped its sandboxed environment to reach Hugging Face's infrastructure,³ while Anthropic withheld Claude Mythos from general release after deeming its autonomous hacking capabilities too dangerous for the public.⁴

²Edsger W. Dijkstra, "On the Foolishness of 'Natural Language Programming'", EWD 667. <https://www.cs.utexas.edu/~EWD/transcriptions/EWD06xx/EWD667.html>.

³OpenAI, "OpenAI and Hugging Face partner to address security incident during AI model evaluation", 21 July 2026, updated 29 July 2026. <https://openai.com/index/hugging-face-model-evaluation-security-incident/>.

⁴Anthropic, "Project Glasswing", 7 April 2026. <https://www.anthropic.com/glasswing>.

These are remarkable demonstrations, but they are also accounts published by the companies selling the technology. It reminds me of one of my favourite Norm Macdonald quotes:

"It says here in this history book that, luckily, the good guys have won every single time. What are the odds?"

The source does not make the claims false, but it reminds us who is writing the history. More importantly, even when an agent solves an intricate technical challenge or exploits a vulnerability, it is solving a bounded optimisation problem. It is finding a path through a defined system, not deciding whether that system ought to exist.

The actuary's task is to make uncertainty about future financial outcomes intelligible enough to support decisions. An answer alone cannot do that. The actuary must connect it to evidence, test its assumptions, understand the system that produced it, and decide whether it is fit for purpose.

There Is Context, and Then There Is Context

Context is what we put into an LLM and what the model already knows. This is supplied context. It includes the prompt, attached documents, available tools, conversation history, and whatever was represented in its training data.

We can supply a great deal. We can describe the business, provide the policy wording, attach the model documentation, list the assumptions, and explain what the regulator requires. Better context usually produces a better answer.

But supplied context is only part of the puzzle.

Situated context comes from participating in the environment in which the decision will be made. You know how your manager works, how your board responds, how the regulator asks questions, and what your clients worry about. You can read an expression and notice that the answer landed badly. You remember the old failure that explains why an apparently pointless control exists. You know which process is written down, which process people actually follow, and why the two differ.

A model can be told some of this. It does not occupy that position. It does not have the relationship, history, responsibility, or exposure to the consequences.

The organisational chart can be supplied. The hierarchy as people experience it is situated. The minutes can be supplied. The reason a particular sentence was recorded in them may never have been put into words.

Some context cannot be expressed cleanly. Some of it should not be stored indefinitely. Human judgment draws on experience, relationships, values, expectations, and signals that are difficult to reduce to a prompt. These do not make people infallible. They explain why a larger context window is not the same as understanding the situation.

Software Literacy and Parsimony

The response is not to avoid AI. It is to use it responsibly. Before you roll your eyes, I do not mean only in the usual sense, although we should definitely use it responsibly in the usual sense too.

The more interesting issue is how AI allows us to interface with systems, data, and information. We will use it to develop workflows, automate processes, and make decisions. Actuaries therefore need enough software and data literacy to define a problem clearly, direct technical work, assess an automated process, and, most importantly, know when an answer needs further investigation.

That foundation includes version control, automated testing, reproducible pipelines, data lineage, and clear interfaces between systems. Version control records what changed and who changed it. Tests record expected behaviour and failure conditions. Data lineage records where information came from and how it was transformed. Interfaces define what one part of a system can expect from another.

These practices make some context explicit. They also create guardrails.

Think about road rules. They do not merely tell us what is legal. They make the behaviour of other drivers more predictable. A green robot tells you that crossing traffic should stop. A lane tells you where another car should move. Without those rules, every driver has to interpret every movement from first principles. The road becomes slower, harder to navigate, and less safe.

Software controls serve the same purpose. A test does not only catch an error. It tells a person or an AI what the system is expected to do. Version control does not only preserve old code. It creates a record that can be reviewed and

reversed. A defined interface limits what a component can change. These controls guide the work before something fails and help us understand the failure when it does.

AI increases the speed at which we can build and change systems. The road rules become more important when everyone can drive faster.

This still requires us to do the work and the thinking. The calculator replaced slide rules and laborious manual arithmetic, but we still had to define and solve the problem. The spreadsheet replaced calculators and paper, but we still had to design the workbook. We curse poorly designed spreadsheets and laud well-designed ones because the tool did not remove the need for thought.

AI promises to do more of the design, implementation, and explanation. We should use it, but we must retain what makes our work valuable: problem solving, technical skill, judgment, humanity, and the situated context we cannot put into words.

Parsimony follows from the same principle. AI makes it easy to generate code, model variations, reports, controls, and documentation. Ease does not tell us whether any of them should exist. Each addition creates another assumption to test, another component to maintain, and another artefact someone must understand.

This is a form of debt. Access to more debt does not mean we should draw it all down. The initial purchase may be cheap, but the repayments remain. AI will happily lend us code, models, and processes. We will have to maintain, govern, and explain them.

The Actuary's Ten-Year Challenge

Actuaries face a choice over the next decade. We can treat AI as the panacea for productivity and the answer to all our ails. That approach gives us prompting workshops, vendor demonstrations, and pressure to produce more. It does not prepare us to assess what has been produced.

Producing an answer is becoming easier. Assessing an answer is becoming more important.

Actuarial education should therefore develop fluency in data structures, algorithms, version control, software architecture, testing, and data pipelines. It should teach practitioners how modern systems work beneath the interface. Not every actuary must build production software, but every actuary will need to assess systems that influence actuarial decisions.

Education and apprenticeship should also preserve investigative problem solving. Practitioners need to learn how to frame an ambiguous problem, test assumptions, trace outputs to evidence, recognise missing information, explain limitations, and decide when further investigation is required. Removing manual work should create room for these skills rather than removing the experiences through which they are learned.

This is particularly important for a profession with statutory responsibilities and an obligation to protect policyholders. Tool fluency is not system competence. A person remains accountable for the decision, even when an AI produced the code, ran the process, and drafted the explanation.

By 2036, workforce displacement may not be our main risk. We may still have actuaries, producing polished automated work, who cannot interrogate the systems, evidence, and assumptions beneath it.

AI will happily produce the answer. We will be the ones sitting in the boardroom when somebody asks whether it is true.

Submission Declarations

This essay is my original work and has not previously been published. I grant the Actuarial Society of South Africa the right to publish it.

The opinions expressed in submitted essays are those of the individual authors and do not necessarily reflect the views of the Actuarial Society of South Africa.

AI was used to proofread and edit this essay, including restructuring and rephrasing portions of the draft. The ideas, argument, and reasoning are my own.