Actuarial Modelling in the Age of AI Agents

How actuarial modelling will be transformed over the coming decade

DANIEL CRAIG RAMSAY

MAY 08, 2025

Share

This article is the second in a series on the topic of AI agents within actuarial wor read the first part in the series follow this link: https://ifoagenai.substack.com/p/ai agents-or-how-i-learned-to-stop

1. Introduction

The emergence of AI agents is set to transform actuarial modelling, presenting unprecedented opportunities to boost productivity throughout every stage of the modelling process. This paper explores the transformative potential of AI agents i actuarial modelling, examining their current applications and considering the far-reaching implications for the future of the actuarial profession. We assess the way which AI agents can support the writing, testing, debugging, and documentation c models, and discuss how the dynamics of human-model interaction may evolve in AI-assisted environment. Furthermore, we draw parallels between the impact of A agents in software development and their anticipated influence in actuarial model positioning recent advances in software automation as a bellwether for the future trajectory of AI-driven automation in the actuarial field.

2. Actuarial Modelling and Software Development Processes: Two Sides of the Same Coin

The actuarial modelling lifecycle shares many similarities with the software development lifecycle. Understanding parallels and divergences between these tw processes is essential to appreciate how advancements in software engineering AI agents could soon be mirrored in actuarial work. In tables 2.1 and 2.2 below we summarise the similarities and differences between actuarial modelling lifecycle a the software development lifecycle. As these tables make evident, actuarial modell can be thought of as a highly specialised instance of software development.

Table 2.1: Similarities Between the Different Stages with the Actuarial Modelling and Software Development Lifecycles.

Aspect	Software Development	Actuarial Modelling
Requirements Setting	Gathering and analysing user needs and functional specifications.	Defining scope, objectives, and data requirements to accurately assess risks and project financial outcomes.
Design Specifications	Creating architecture and detailed design documents.	Specifying model structure, assumptions, and methodologies to be used.
Coding and Development	Writing code to build applications.	Writing code to represent model logic.
Testing and Validation	Conducting unit tests, integration tests, and user acceptance tests.	Performing validation through replication spreadsheets, stress testing, and sensitivity analysis.
Documentation	Writing user manuals, API documentation, and code comments.	Writing documentation covering model design, assumptions and data sources.
Maintenance and Updates	Updating software to fix bugs, add features, and adapt to new technologies.	Periodically updating model to incorporate new data, adjust assumptions, and comply with regulatory changes.
Interaction with Systems	Integrating with other systems through APIs or data feeds.	Interacting with various data sources, financial systems, and reporting tools.

Table 2.2. Differences Between Actuarial Modelling and Software Development.

Aspect	Software Development	Actuarial Modelling
Output and Deliverables	Functional applications or systems used by end-users.	Financial reports and projectio used to analyse risks and forec future scenarios.
User Expertise	Coding generalists with knowledge of formal software processes and technology architecture.	Coding specialists with focus on and financial modelling.
Regulatory and Compliance	Compliance with security standards; not typically subject to stringent regulatory scrutiny.	Often subject to stringent regula and compliance requirements necessitating high accuracy, transparency, and auditabilit

This evidence of accelerating AI adoption in software engineering provides a valu lens through which to consider the likely trajectory of actuarial modelling over th coming decade. Indeed, actuarial modelling can be seen as a specialised branch of software development, sharing core principles of algorithm design, data manipula and automation—though tailored to domain-specific requirements. Innovations ir modelling tools and automation have tended to emerge first in mainstream softwa development, with actuarial practice often adopting these advances with a noticea lag, sometimes as much as a decade later. Figure 2 illustrates this close relationshi highlighting parallel evolutionary paths, while also showing how advancements ir general software development frequently precede those in actuarial modelling. As autonomous AI agents increasingly shift the focus in software from manual codin orchestration and high-level system management, we should expect comparable transformative effects in actuarial modelling. Actuaries are thus likely to benefit f the productivity and quality gains pioneered by their counterparts in software engineering, as AI agents become more deeply embedded in actuarial workflows. However, the pace of this change may well surpass that of previous adoption cycle underscoring the importance for actuarial professionals to proactively engage wit emerging technologies.

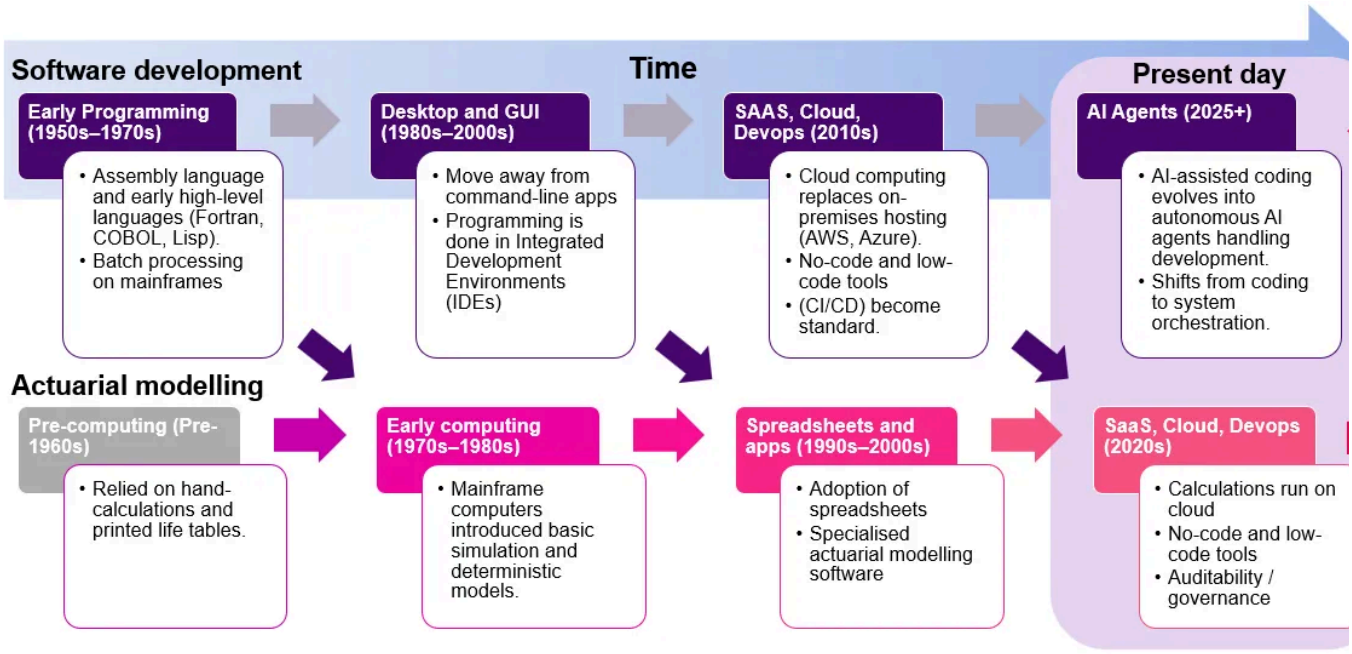

Figure 2. Actuarial modelling is typically 10 or more yea behind software engineering practices. Improvements ir the capabilities of AI agents in the context of software development will soon have influence on how actuaries develop models.

3. The Impact of Generative AI on Software Development

GenAI is rapidly transforming the software development industry, altering how developers write code, debug, and maintain software. Through leveraging powerfu models like GitHub Copilot, ChatGPT, and other GenAI coding assistants, softwa development processes are becoming materially more efficient and less time-consuming.

According to research conducted by GitHub (Github 2022), developers using Githu Copilot reported higher job satisfaction, reduced frustration, and an increased abi to focus on more satisfying tasks. Specifically, between 60% of users felt more fulfi in their roles, 96% reported that Copilot helped them complete repetitive tasks fas and 87% stated it conserved their mental effort during repetitive tasks. Additionall

controlled experiments, developers using GitHub Copilot completed tasks 55% fas on average compared to those who did not use the tool, with a success rate of 78% versus 70%. Indeed, several other studies have reported similar findings in regards the productivity gains achieved with LLM-based tools (see Table 3).

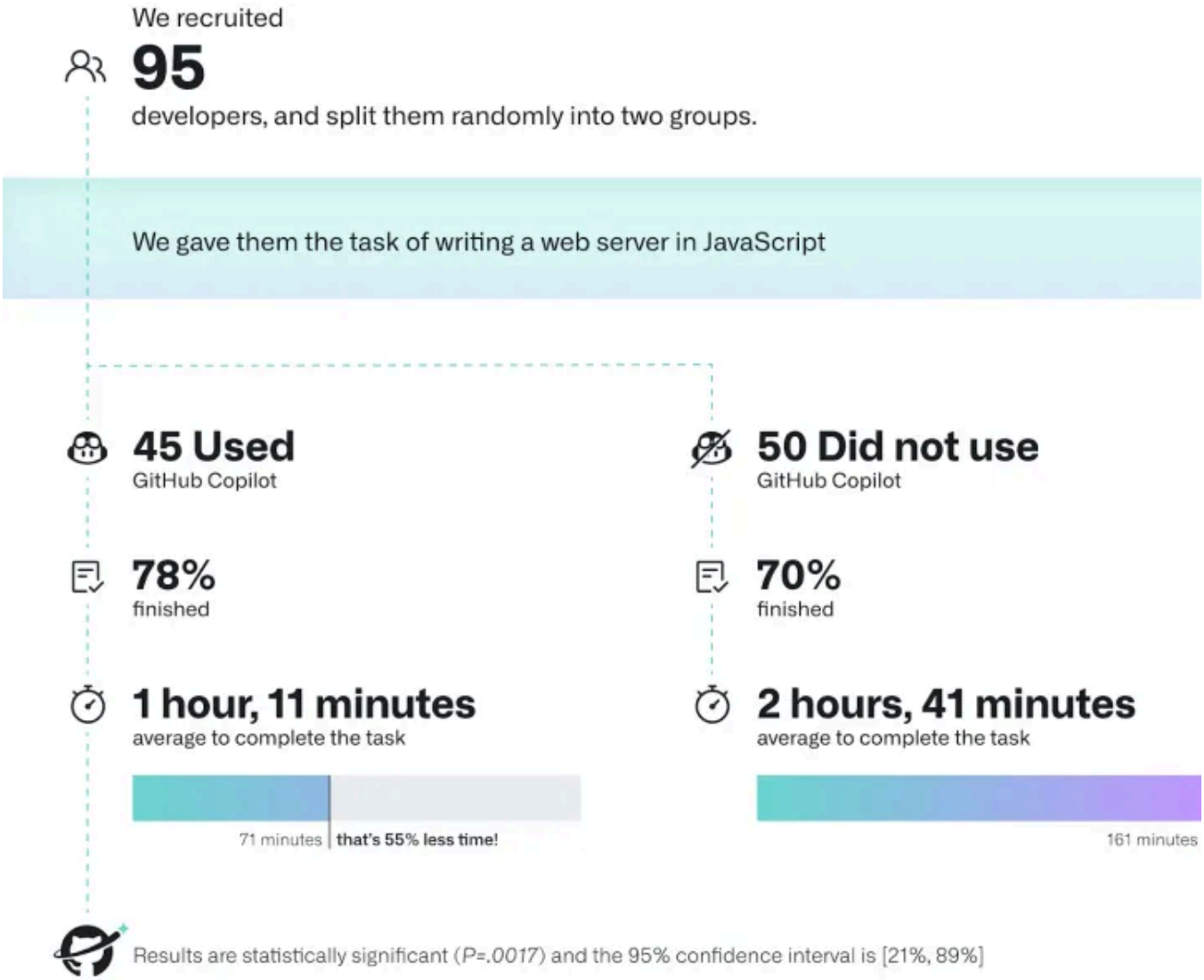

Figure 3: Summary of the Impact of GitHub Copilot on Developer Productivity. Source Microsoft Research (202

Table 3: Summary of Key Studies on LLM Impact on Developer Productivity.

Study	Methodology	LLM/Tool Used	Key Metrics Measured	Key Findings
Microsoft Research GitHub Copilot Study (Microsoft Research, 2023)	Controlled Experiment	GitHub Copilot	Task Completion Time, Task Success	55.8% faster task completio
Turing Study (Turing, 2025)	Field Experiment	Turing AI Assistant, Cloud-Native Assistant	Successfully Reviewed and Merged PRs	33% increase i PRs
Ant Group Study (Ant Group, 2023)	Field Experiment	CodeFuse	Lines of Code Produced	55% increase i LOC (primarily junior staff)
McKinsey Study (McKinsey, 2023)	Analysis of Findings	Generative AI Tools	Coding Task Completion Time	Up to twice as fast completio

These productivity gains are impressive, but it is important to note that LLMs hav improved substantially in terms of coding ability since the publication of the resul these studies. Hence we can only conclude that the gains are likely far higher at th time of writing (May 2025).

Building on these early advances, the trajectory of AI in software engineering has progressed from simple code-suggestion copilots to fully agentic systems capable managing complex development workflows—exemplified by emerging agentic ID like Windsurf and Cursor.. As outlined in Part 1, one of the best indicators of this evolution is the SWE-bench benchmark, which tests how well AI can autonomous resolve real-world software issues. When SWE-bench was first deployed, general-purpose models such as ChatGPT 3.5 could solve only about 0.4% of these tasks—underscoring the clear limitations of even impressive early LLMs for full-cycle engineering.

Since then, agentic architectures employing explicit multi-step planning, tool-use reasoning have brought rapid advances. OpenAI's o3 model, released in Decembe 2024, surpasses a 70% verified solution rate on SWE-bench and is approaching superhuman-level proficiency, with leaderboard performance now flattening as th benchmark reaches saturation. This jump, observable in less than one year, eviden not only the extraordinary pace of capability growth, but a fundamental transformation in how software will be built, tested, and maintained.

This revolution is visible in a wave of tools moving far beyond simple code completion:

- **GitHub TestPilot** (GitHub Next, 2025a) can automatically generate comprehe unit tests, radically accelerating the quality assurance cycle.
- **GitHub Copilot for Pull Requests** (GitHub Next, 2025b) leverages AI to draft detailed explanations and commentary when merging code, streamlining the process and improving documentation and auditability.
- **SpecLang** (GitHub Next, 2025c), an experimental tool, empowers developers t author and refine applications through structured natural language specificat with the agent autonomously generating and updating executable code as requirements change.

The rise of such agentic support is fundamentally changing the division of labour software engineering. Rather than focusing on rote implementation and manual debugging, developers are freed to concentrate on creative design, system architec and higher-order problem solving, while agents handle the scaffolding, testing, an documentation. This shift is not just about speeding up coding; it is a redefinition the role of software developer.

The rapid progress chronicled in evolving benchmarks like SWE-bench—now nea saturated for some leading models—shows that the discipline is at a critical inflec point. A corresponding transformation in actuarial modelling is likely only a few y behind. As discussed previously, the structure of actuarial model development clo mirrors that of software engineering: model specification, implementation, testing documentation, and reporting (see Table 2.1). The very same principles and agenti

tools revolutionising coding are poised to automate large portions of the actuarial process as well.

4. Automating Actuarial Modelling Tasks with A Agents

We should therefore anticipate that actuaries, just like software developers, will sc shift toward oversight, high-level specification, model governance, and strategic innovation. Routine or repetitive processes—data ingestion, formula translation, regression testing, and documentation—will increasingly fall to AI agents designe the actuarial domain. AI agents will become embedded in various aspects of the modelling process beyond code generation, enabling actuaries to focus on more complex and value-added tasks. Table 4 below outlines how specific tasks within t actuarial modelling process could evolve in this new AI-augmented environment.

Table 4: Deployment of AI Agents within the actuarial modelling life cycle.

Task	Role of AI Agent
Converting Requirements	Agents transform high-level requirements into detailed first draft model specifications for a human actuary to review
Developing and Implementing Models	Agents autonomously generate entire solutions based on these model specifications rather than generating code bases on a line-by-line basis
Data Validation and Cleansing	Agents will automate data validation and cleansing through autonomously identifying issues with the data and generating appropriate cleansing rules to be applied
Generating Test Cases and Debugging	Agents will generate comprehensive sets of test cases and autonomously run these tests and debug and correct failing tests iteratively until all tests pass
Documentation Generation	AI agents will generate detailed documentation for all parts of the modelling solution artefacts they generate ensuring model transparency and regulatory compliance
Reporting and Analysis	AI agents could automate report generation and formatting, enabling actuaries to focus on interpreting results and decision-making.

AI agents will not only be capable of performing various modeling tasks, but they also likely to do so with greater reliability and quality. This is because AI can be instructed to ensure that the actuarial model artefacts they generate are always maintainable and fully comply with necessary best practices and regulatory standa AI may achieve a level of consistency and precision that surpasses human actuarie

who, constrained by time pressures or inattention, may sometimes be forced to or inadvertently cut corners.

5. The Future of Human-Computer Interaction in Actuarial Modelling

5.1. Models of Human and Agent Engagement with Applications

Human-Computer Interaction (HCI) encompasses the ways in which humans enga with computers and software, as well as the design of systems that support and enhance these interactions. As AI agents become increasingly embedded in actua modelling, the dynamics of this interaction are set to undergo substantial transformation. In this context, we examine four potential scenarios outlining hov actuaries' engagement with their models may evolve beyond the conventional desk based approach.

Table 5.1: Emerging Models of Human and Agent Engagement with Applications.

Category	Description
0. Traditional Desktop-Based Approaches	In the traditional setup, all deliverables—such as inputs, assumptions, model code, tests, and documentation- manually modified by the user. This approach is entirely human-driven, with each component requiring direct u input and management.
1. App-Specific AI Copilot-Based Systems	These are systems which have AI features but those features are limited in their ability to perform actions withi application directly, with the copilot able to answer questions and produce code but not run models or analyse r
2. App-Specific AI Agent-Based Systems	In this scenario, both the human user and an AI agent—tailored specifically to the modelling platform—are abl create and modify all project deliverables. The agent is tightly integrated with the software, functioning similarl generative AI copilots increasingly found in enterprise tools, but with the added capability to write to the model context. A key advantage is that domain-specific knowledge can be embedded directly into the agent, reducin reliance on the more generalised knowledge of a pre-trained, platform-agnostic model.
3. Generic AI Agent-Based Systems	In this scenario, both the human and a generic external AI agent can create and edit all modelling project deliverables. The AI agent operates externally to the modelling software, controlling the browser or operating s that hosts the application. This setup is akin to OpenAI's Operator model, offering flexibility for using applicatic features and allowing direct file manipulation. However, generic agents may lack actuarial-specific knowledge might require continual feedback. They may also struggle with bespoke or proprietary applications without comprehensive documentation or human guidance.
4. External and App-Specific Hybrid Systems	In this multi-agent scenario, a human can manually edit and view all files in a modelling solution while delegati tasks to an external AI agent. The external agent can also interact with a specialised actuarial modelling agent embedded within the software. This hybrid system combines the flexibility of external AI interaction with the specialised knowledge of an app-specific AI agent fine-tuned for actuarial modelling solutions.

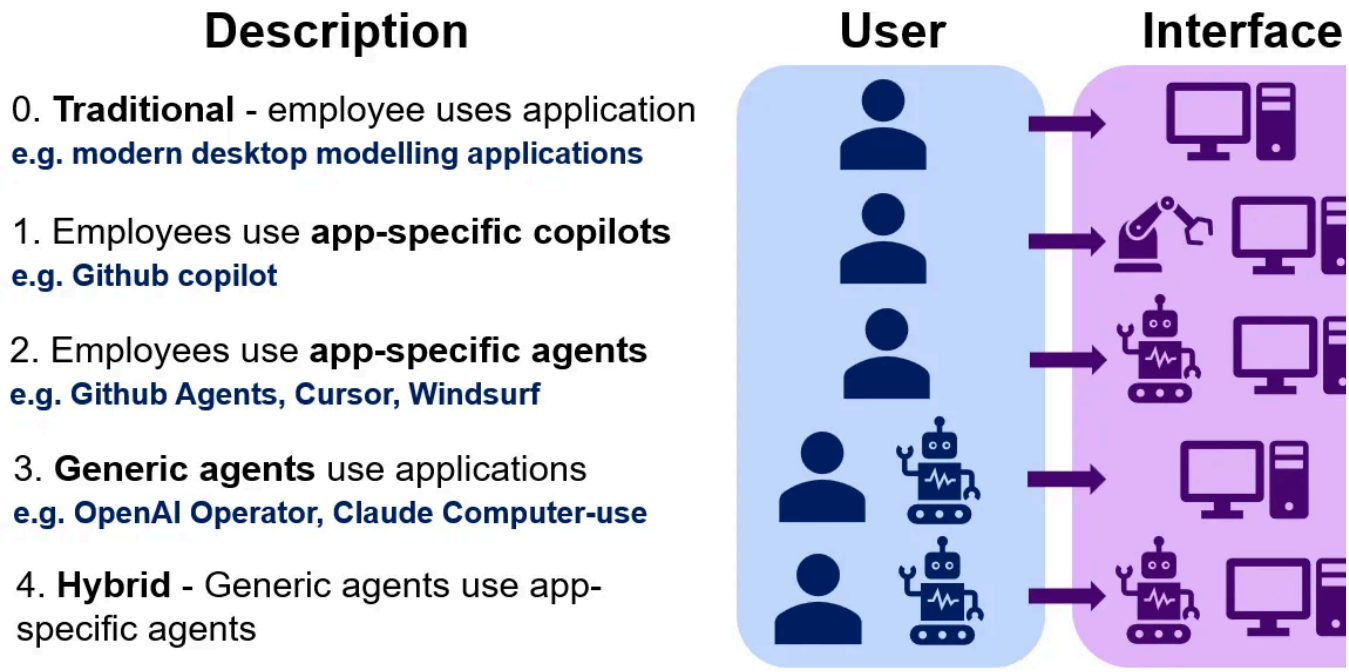

Figure 5.1: Summary of Emerging Models of Human and Agent Engagement with Applications.

Ultimately, these emerging models signal a profound transformation in how we interact with computers in our professional lives. As AI agents assume a growing

share of day-to-day modelling and analysis tasks, the traditional role of the actuar a "doer" will evolve towards that of a reviewer or overseer—where the primary responsibility shifts from constructing models manually to validating outputs and exercising final judgement. For analysts, whose roles often involve repetitive, structured work, this will be particularly impactful, as many of their core activitie stand to be delegated to AI agents. In contrast, senior professionals, who focus mo on interpreting management information, strategic decision making, and exceptio based oversight, will see comparatively less change in their day-to-day function in near term. This emerging paradigm will fundamentally reshape our relationship w workplace technology, pushing more routine tasks toward automation and placing premium on human expertise in supervision, critical review, and decision-making.

5.2. Architectures of agentic systems

Another important consideration in the design of app-specific agent scenarios is t architecture of AI agents within a modeling solution. Broadly, such architectures c be categorised as: monolithic, agent-orchestrated multi-agent systems, or hierarch multi-agent systems.

- In a **monolithic system**, a single agent is responsible for all aspects of plannin and tool use required to fulfil a task, see Figure 5.2.1.
- In contrast, an **agent-orchestrated multi-agent system** comprises several specialised agents, each focused on a particular task or domain. In an actuaria multi-agent system, each individual sub-agent might be dedicated to one spec task within the modelling lifecycle like coding, testing or documenting. These agents operate largely independently and typically require initiation or prompt from a human modeller to perform tasks.
- Alternatively, a **hierarchical multi-agent system** features a lead agent that oversees the overall process and delegates specific responsibilities to various s agents. These sub-agents may themselves delegate tasks to more specialised c agents, resulting in a multi-layered, scalable structure for task management. F instance, in an actuarial hierarchical system, a top-level agent could coordinat

the entire modelling workflow, while a Manager sub-agent might assign tasks code writing and code review to lower-level child agents.

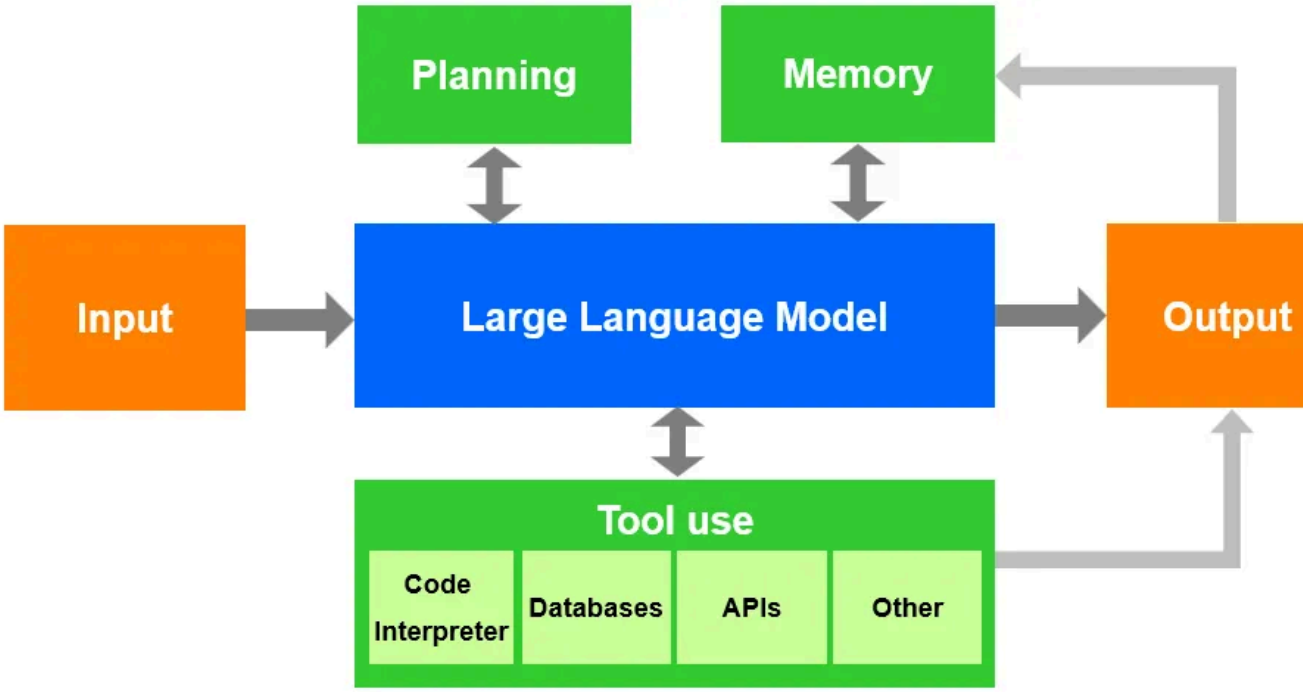

Figure 5.2.1. A monolithic agent (or individual nodes within a multi-agent) system can have its own capacity f planning, memory and tool use.

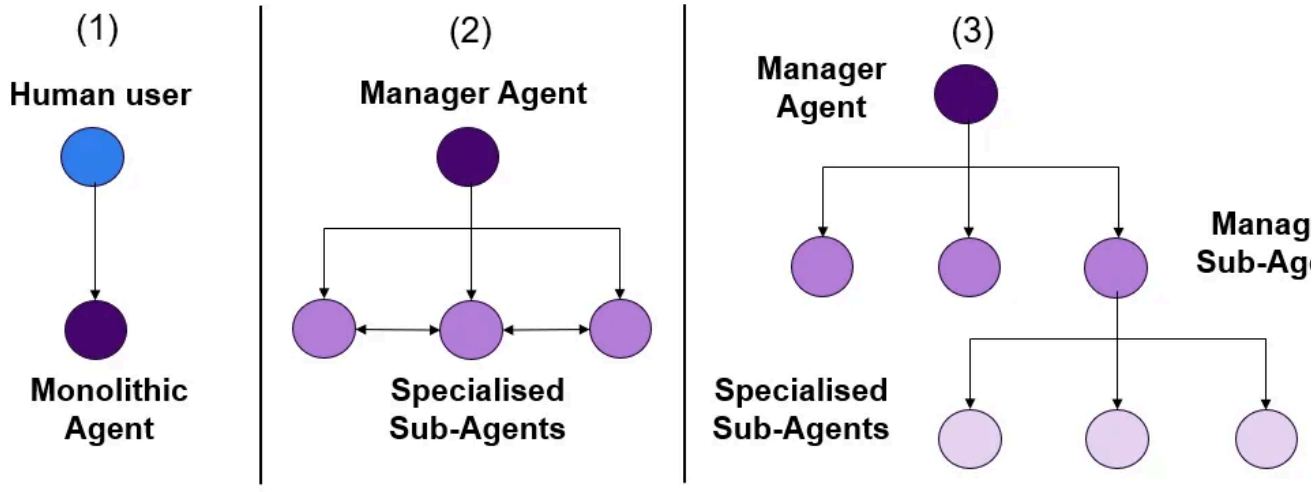

Figure 5.2.2. (1) Monolithic System (2) Human Orchestrated Multi-Agent System (3) Hierarchical Agent Orchestrated Systems.

This spectrum of agent architectures offers varying degrees of specialisation, autonomy, and scalability, each bringing distinct advantages depending on the spe requirements of a solution. Monolithic agents are simpler to architect and would l serve as the first iteration of an agentic system. However, multi-agent systems pro important benefits: they can be more token-efficient, thereby reducing compute c and they distribute responsibilities among agents, lowering cognitive load and improving reliability. When a single agent is tasked with too many responsibilities attention mechanisms can become overwhelmed, which may compromise performance. By dividing tasks, multi-agent and hierarchical systems help ensure more stable and effective operation. A further advantage of multi-agent approache their ability to operate asynchronously and in parallel, significantly boosting effici and speed. Modern frameworks such as LangGraph (2025) and CrewAI (2025) now make the development of such sophisticated multi-agent systems increasingly accessible.

It is important to note, however, that the adoption of these AI-driven architecture does not necessarily eliminate the role of the human modeller. All artifacts genera by the AI agents can remain accessible for review and modification by humans. Moreover, the agents should ideally be designed so their reasoning and modeling choices are clearly documented within logs or accompanying documentation to en transparency within the process.

It is possible that the architecture of human–agent interaction will gradually evolv through various stages over the coming years. Alternatively, a single paradigm ma emerge as the dominant approach, depending on how the field develops. Much of evolution will be closely tied to advances in the intelligence and reliability of AI systems.

As AI becomes more capable and trustworthy, it will be feasible to delegate increasingly abstract tasks and higher-level control to AI agents—moving beyond current state, where significant scaffolding and human oversight are required to ensure sufficient accuracy and reliability. It is anticipated that the range of tasks assigned to AI agents will continue to expand. In turn, their influence within the

modeling process is likely to grow, transitioning from isolated, task-specific suppo more holistic management and oversight of the entire modeling workflow.

6. A Future with AI-Enhanced Actuarial Modellers

6.1. A Historical Perspective

Reflecting on the 1990s, actuaries might have feared that the advent of modelling software would automate away their Excel-based roles. However, the profession ha only grown, with technology eliminating low-level manual work and enabling actuaries to undertake higher-level, strategic tasks. In a regulatory environment th favours increasingly complex internal models or pricing systems, AI agents will lik create capacity for actuaries to develop even more sophisticated models that will require yet more actuarial oversight to ensure they meet financial commitments o insurers and pension funds. Hence, rather than diminishing actuarial roles, AI cou potentially augment the profession, increasing the demand for actuaries to oversee more complex modelling solutions.

6.2. Implications for Low-Code/No-Code Solutions

However, a counterpoint to this rather optimistic conclusion that increasing complexity increases the demand for actuaries is that a significant portion of actu modelling currently relies on low-code/no-code solutions due to a shortage of skil developers. In a world where development skills are scarce, a single highly skilled developer can only scale their modelling output by managing a team of low-code developers. However, if AI enhances the productivity of these rare highly skilled developers, the reliance on low-code solutions will likely decrease. In this scenario one could argue the demand for highly skilled developers will remain high, as thei amplified productivity allows them to handle more complex projects independentl consequence of this is that actuaries whose lesser technical abilities are currently

accommodated by low-code/no-code solutions may find their roles becoming less necessary. Such individuals could benefit from enhancing their coding abilities to remain competitive, ensuring they are augmented rather than displaced by AI-led automation.

6.3. Leveraging Human Strengths in Actuarial Modelling and The Risk of AI-Generated-Complexity

While AI can automate numerous actuarial tasks, human expertise remains crucia complex, strategic, and interpretative work. Complex models need to be well-structured and comprehensible to humans, not only for internal use but also for regulatory compliance and stakeholder transparency. Even as AI crafts models of increasing complexity, actuaries will still be needed to ensure these models remai transparent and justifiable.

A potential risk of incredibly sophisticated AI systems is that they are able to crea models so complex that they become unjustifiable or too incomprehensible to hun actuaries. For instance, an AI might determine that writing an actuarial model in machine code or assembly language offers significant efficiency gains over higher-level languages like Python. While this might be computationally advantageous, it raises concerns about human interpretability and trust. Even if the AI can docume these models, actuaries must be able to verify that the models requirements are m

6.4. Limitations of AI Agents and the Imperativ for Good Model Design

While the promise of AI agents in actuarial modelling is significant, it is importar recognise their inherent limitations and the challenges involved in their practical deployment. AI agents are not a panacea for the resource constraints faced by mar actuarial teams, nor should their adoption encourage greater model complexity fo own sake. The primary purpose of actuarial models—regardless of technological

innovation—remains to deliver clear, consistent, and replicable outputs that are ea understood and validated.

One temptation as AI capabilities advance is to build ever more elaborate modelli systems, leveraging stochastic methods and machine learning in pursuit of margir predictive gains. Yet, experience in both actuarial science and software engineerir shows that transparency, simplicity, and maintainability are often of greater value intricate, opaque designs. The ideal use of AI agents is not to complicate the modelling process, but instead to automate clearly defined, deterministic calculati layers—thereby freeing actuaries to focus on judgement and oversight rather than computation. Actuarial agentic systems should be designed with a deterministic "spine"—grounded in rule-based logic and well-understood actuarial principles.

Another central challenge in building reliable AI-driven actuarial systems is the codification of best practices. Unlike software engineering, which benefits from a broad, publicly accessible set of standards, libraries, and conventions, much of actuarial modelling expertise resides in the tacit knowledge of experienced practitioners. These hard-won insights, built up over years, remain largely undocumented and are often shared informally within teams or organisations. Translating this embedded expertise into system prompts for instructing agents is non-trivial task—yet it will be crucial for implementing an agentic system with an true utility. Without explicit, standardised guidance, there is a risk that agents wil replicate only surface-level aspects of actuarial coding, missing the deeper rationa required for robust, audit-ready modelling.

To fully benefit from the integration of AI agents, the actuarial profession may ne take inspiration from the discipline of software engineering—treating actuarial modelling as a craft worthy of rigorous study and documentation in its own right. would involve systematically articulating not only technical methods but also the l standards of clarity, reproducibility, and governance that define actuarial work. Or by capturing the practices and unwritten rules developed by generations of actuar and embedding them into automation frameworks can we ensure that AI augment rather than undermines, the foundational principles of the profession.

As AI agents become a more central part of actuarial modelling, it will be vital to emphasise “minimum-viable complexity”—retaining transparency, reproducibility and ease of validation as non-negotiable requirements. Actuaries and technology leaders must remain vigilant against the allure of complexity for its own sake, and thinking about how transfer of actuarial knowledge to AI-driven systems should b achieved. Ultimately, the goal should be not just greater automation, but better, m accessible models—anchored in the trust and expertise that have long been the hallmarks of actuarial science.

6.5. Closing Remarks

The emergence of AI agents in the actuarial modelling sphere signals a transform future for the profession. To thrive in this evolving landscape, actuaries must not o embrace these technological advancements but also foster a culture of innovation thoughtful implementation.

As AI becomes more integrated into our professional lives, the role of actuaries wi expand beyond traditional programming. New skill sets—such as prompt enginee AI model evaluation, AI governance, and managing AI-enhanced teams—will bec increasingly essential. These capabilities will enable actuaries to build and oversee models of far greater complexity with less manual effort.

Yet, even in an AI-augmented environment, the human element remains indispensable. Judgment, strategic thinking, and oversight will continue to define profession's value. History has shown that technological progress does not elimina actuarial roles—it elevates them. In the future, actuaries will work alongside AI, pushing the boundaries of what is possible in modelling and shaping a more dyna data-driven profession.

References

- Ant Group (2023) We investigate the impact of Gen AI on labour productivity through a field experiment in the coding industry. Available at:

https://www.bis.org/publ/work1208.htm.

- ARK Invest (2024) 'A Revolutionary Blood Test Could Lead To Predictive And Prescriptive Analytics, & More'. Available at: https://www.ark-invest.com/newsletters/issue-444.
- CrewAI (2025) CrewAI. Available at: https://www.crewai.com/
- GitHub (2022) Research: quantifying GitHub Copilot's impact on developer productivity and happiness. Available at: https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/.
- GitHub Next (2025a) GitHub TestPilot. Available at: https://githubnext.com/projects/testpilot/.
- GitHub Next (2025b) Copilot for Pull Requests. Available at: https://githubnext.com/projects/copilot-for-pull-requests/.
- GitHub Next (2025c) SpecLang. Available at: https://githubnext.com/projects/speclang/.
- LangGraph (2025) LangGraph. Available at: https://www.langchain.com/langgr
- McKinsey (2023) Unleashing developer productivity with generative AI. Avail at: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai.
- Microsoft Research (2023) The Impact of AI on Developer Productivity: Evide from GitHub Copilot. Peng, S., Kalliamvakou, E., Cihon, P. and Demirer, M. 1 Microsoft Research. 2 arXiv. Available at: https://doi.org/10.48550/arXiv.2302.06590.
- OpenAI (2025) 'Introducing OpenAI o3 and o4-mini', OpenAI. Available at: https://openai.com/index/introducing-o3-and-o4-mini/.
- SWE-bench (2024) SWE-bench: Can Language Models Resolve Real-world Git Issues? The Twelfth International Conference on Learning Representations. Available at: https://openreview.net/forum?id=VTF8yNQM66.
- Turing (2025) LLM Coding Assistants Increase Software Development Productivity. Available at: https://www.turing.com/resources/llm-coding-

assistants-increase-software-development-productivity.

- VentureBeat (2024) ‘OpenAI confirms new frontier models o3 and o3-mini’. Available at: https://venturebeat.com/ai/openai-confirms-new-frontier-models and-o3-mini/.

Discussion about this post