The Safety-Critical Systems Club Newsletter

Safety Systems

Vol 34 No. 3 – Oct 2026



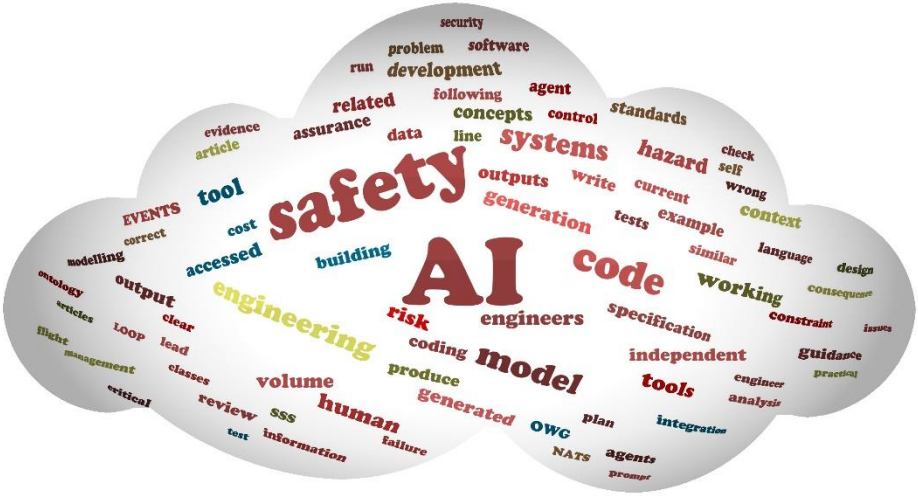
CAMBRaiN EXPLOSION
Evolution in AI Primordial Seas

KEPT IN THE LOOP?
How to Assure AI Outputs

DEVIL IN THE DETAIL
Losing Our Way(point)

SCSC

thescsc.org



SCSC Publication Number: SCSC-213

While the authors and the publishers have used reasonable endeavours to ensure that the information and guidance given in this work is correct, all parties must rely on their own skill and judgement when making use of this work and obtain professional or specialist advice before taking, or refraining from, any action on the basis of the content of this work. Neither the authors nor the publishers make any representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability or availability with respect to such information and guidance for any purpose, and they will not be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever (including as a result of negligence) arising out of, or in connection with, the use of this work. The views and opinions expressed in this publication are those of the authors and do not necessarily reflect those of their employers, the SCSC or other organisations.

© SCSC CIC 2026 except where explicitly stated for specific articles, reports or images.

Front cover: AI generated image
In Brief images may be subject to copyright – contact the SCSC Newsletter Editor for details.

Contents

WELCOME

Editorial 3

Opening words from the SCSC Newsletter Editor.

In Brief 4

Recent system safety news items from around the world.

FEATURES

The CambrAIn Evolutionary Explosion 5

Paul Hampton provides a comparison between the Cambrian evolutionary explosion and the present acceleration of AI-assisted software development tools.

I don't want to spend the rest of my career reviewing AI outputs 15

Ben Fulford discusses how to manage the overwhelming amounts of AI-generated outputs.

Devils in the Detail 23

The Ontology Working Group (OWG) conclude their series of articles on ontological modelling.

SCSC Safety Systems Crossword 44

The annual SCSC Safety Systems crossword returns for its 5th year!

REPORTS

How to Make the Most of AI: Seminar Report 35

Catherine Menon provides highlights of this AI-related seminar and workshops

60 Secs with ... Tim Rowe 42

Tim answers some quick-fire questions on system safety and the future.

GROUPS

Working Groups 47

Details of the SCSC Working Groups.

SCSC Steering Group 56

Who's who in the Steering Group.

EVENTS

Calendar 58

Events Diary 59

SCSC

For Everyone Working in System Safety

SSS'27 Programme



35th Safety-Critical Systems Symposium

9 - 11 February 2027

Bristol Harbour Hotel, Bristol, UK

scsc.uk/sss

thescsc.org/sss

Editorial

If you were at the SCSC Symposium in February this year you will have hopefully caught the technical entertainment where we dabbled with some of the interesting things we could do by asking AI tools like ChatGPT and Gemini to write software for us. We might have laughed at its attempt to generate a fault tree from a document, but the code generation was impressive.

I was asked to talk about AI code generation for the June 2026 seminar How to Make the Most of AI: A Workshop for Safety Engineers (scsc.uk/e5101) and while researching the talk I was simply blown away with the current capability with complete functioning and tested applications being written in the space of minutes from one-page specifications.

I've since been exploring over 20 different projects from websites, Android apps, ontological modelling to prose generation and the capability is simply incredible. From my 35 years' experience in the industry, and most of that in software development, I've not seen anything like this and I truly believe we are now in a new industrial revolution. I've taken the liberty to write an article on the subject 'The CambrAIn Evolutionary Explosion' and hopefully that will explain how monumental this period is and how rapidly things are changing. I wrote the article originally in June and it needed several significant revisions now in October; the pace of movement is extraordinary and it's hard to keep up. As you might guess, the article is heavily supported by AI-based research; it is still my intent, my vision and style, but AI has greatly helped in its production. It's still classed as my work (ask AI if its AI-generated and it will say no) but the AI contribution has been fundamental and a huge accelerator. It's hard to see a world now where we're not using AI in some capacity.

AI has its critics of course, and I am perhaps on the hugely positive side of things! Our second article is from Ben Fulford who was a co-presenter at the June seminar. Ben also takes a very positive stance on the use of AI and is starting to think about where it will all lead – surely we don't want to spend our careers checking AI outputs, which will become increasingly hard to find error in?

The Ontology Working Group (OWG) also provides its fourth episode on its journey to explain ontological modelling techniques and the benefits these bring to everyone. The modelling draws from the issues surrounding the NATS air navigation system outage in 2023 and so makes interesting reading about how those issues arose.

Catherine Menon kindly provides a report of the June AI Seminar. The video recordings and all of the presentation slides and tutorial material are available for download to SCSC members. There are still three further SCSC seminars in the coming months, the first two on AI in safety workflow and AI assurance and the third on new techniques in Hazard Identification. The annual symposium in February 2027 (scsc.uk/sss), is shaping up to be a fantastic event now with three parallel tracks throughout the three days, offering a choice of over 65 talks, workshops and tutorials on a systems safety theme. We are also in a new marvellous venue of the Harbour in Bristol and I really hope you can make it!

In this issue we also have our annual prize Safety Systems Crossword so try your hand to win a prize!

Our 60 second interview is with Tim Rowe, an experienced safety consultant, former NATS air traffic engineer and expert embroiderer!

Paul Hampton
SCSC Newsletter Editor
paul.hampton@scsc.uk



In Brief








Failed engine component and an ambiguous warning caused fatal Royal Navy Merlin helicopter ditching



Two years after a Merlin HC Mk 4 flying from HMS Queen Elizabeth ditched in the sea off Dorset, killing Lt Rhodri Leyshon, the board of enquiry report has finally been published.

The inquiry found that a fractured engine component triggered an emergency that the crew misdiagnosed, after which two serviceable engines were shut down, leaving the helicopter without power at low level over the sea at night. *navylookout.com*



More flight cancellations as fallout from air traffic control glitch continues

A major technical failure at National Air Traffic Services (NATS) on 8th Sep 2026, caused widespread travel chaos across the UK. A system glitch at the Swanwick control centre disrupted flight plan processing, forcing air traffic controllers to halt departures.

The outage led to over 1,750 flight cancellations and affected more than 300,000 passengers across major airports. *bbc.co.uk*

Will self-flying planes transform the skies?



The race to deploy autonomous, fixed-wing aircraft is accelerating, led by pilotless crop dusters and cargo delivery planes rather than urban air taxis. California-based start-up Pyka has secured US commercial authorization for its fully electric, autonomous crop sprayer, which flies lower than human pilots to reduce chemical drift.

Meanwhile, British firm Windracers is seeking permission for heavy-lift cargo services. While autopilot merely assists, fully autonomous systems handle entire flights via software algorithms. However, strict aviation safety standards have slowed deployment compared to self-driving cars. *bbc.co.uk*



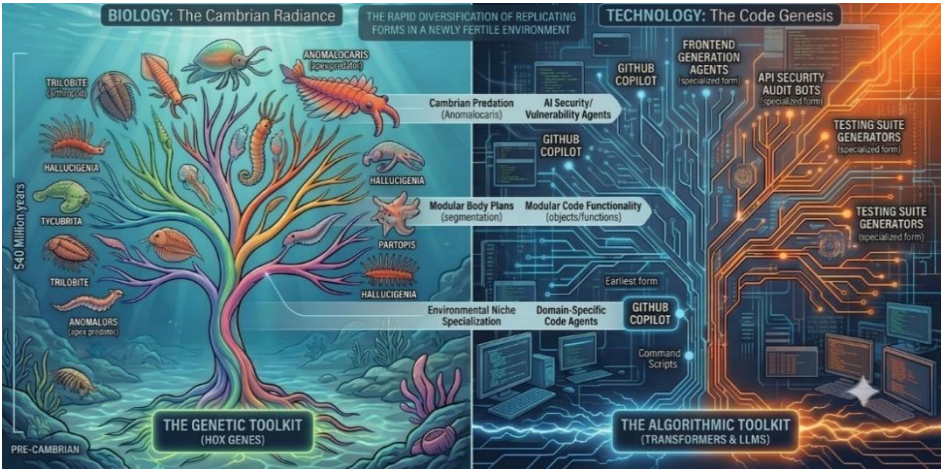
AI to Reroute Atlantic Flights to Cut Warming Contrails

A £5 million UK project is turning to artificial intelligence to help passenger jets dodge the climate-warming condensation trails left in their wake.

Known as Operation Blue Skies, the 30-month initiative involves Google, the UK government, NATS, and the Met Office. It targets Shanwick airspace in the eastern North Atlantic, an area responsible for roughly 5% of global contrail warming.

Experts estimate these clouds account for up to a third of aviation's total climate impact. *bbc.co.uk*

The CambrAIn Evolutionary Explosion



Paul Hampton likens the acceleration of Artificial Intelligence (AI) assisted software development tools to the Cambrian biological evolutionary explosion. Paul argues that the many tools on the market are not merely improving programming productivity, but changing the ecology of system creation itself. Paul examines market turbulence, the shift from in-editor assistance to persistent agents, rapid capability growth and the implications for safety-critical engineering.

An Analogy of Sorts

Over 500 million years ago, the Earth experienced an unprecedented and seismic change in the development of complex animal life called the Cambrian Explosion [1]. During this period, the fossil record shows a rapid expansion of complex animal life. Where previously, the primordial seas were home to soft-bodied species like jellyfish and sponges and other more basic lifeforms such as bacteria, the period saw the emergence of a vast array of new different species, and in fact, many of the major animal body plans represented today emerged during this period. A fundamental change was the development of hard mineralised skeletons and the earliest vertebrates – game changers in securing dominance in the evolutionary battlefield where trilobites and fish, amongst others, led the charge. And a battlefield it was, with new forms appearing, competing, specialising and disappearing.

I strongly believe that we are now experiencing an analogous explosion in the domain of AI assisted development environments, which seem to be emerging, multiplying and mutating with developers, enterprises, regulators and commercial markets acting as the selectors for their evolution. This will now drive a rapid expansion of applications through vastly accelerated development and the democratising of code development, where literally anyone, with a little technical setup, can now develop applications.

A New Dawn

In advance of my talk at the SCSC Seminar "How to Make the Most of AI: A Workshop for Safety Engineers" [2] in June this year, I decided to research AI code generation capabilities and started my journey by simply using ChatGPT, Claude and Gemini to review, fix and generate code. The results were remarkable with the tools *seemingly* displaying an intelligence about my objective and a guru level of language and software engineering knowledge. I was then introduced to Anysphere's Cursor [3] – an Integrated Development Environment (IDE), which took the capability to an entirely new level.

Workflows that once required weeks of scaffolding, experimentation, interface construction and error correction could now be compressed into a matter of days, and sometimes into a single sustained session. I could now write fully functional and tested applications with graphical interfaces in tens of minutes; mobile applications that might take at least 3 months to write, test and debug took a couple of days. Insights into previously difficult coding issues and problems could be resolved in a matter of seconds saving hours of head scratching.

A couple of workshops at the seminar demonstrated the capability. In timeframes measured in minutes, Cursor generated two working and tested applications including web-based user interfaces from pre-prepared single page specifications:

- A validator for Airbus A320 pitot airspeed sensors using a 2oo3 (two-out-of-three) software architectural pattern for three independent airspeed sensors deployed on the aircraft, providing safety-critical aircraft speed information to the flight control system
- A Monte Carlo simulation (see image opposite) to predict the safety margins of an Autonomous Train's braking system. This would run for the thousands of selected iterations taking a number of fixed values as well as assumed statistical distributions of other variable parameters



Monte Carlo simulation for a automated train braking system. Tested software generated by Cursor in minutes from 1 page spec.

The more capable tools now combine Large Language Models (LLM), repository indexing, command execution, test running, documentation search, code review and conversational planning. GitHub Copilot began as an assistant for completing lines and functions, but now spans chat, agent mode, issue to proposed code fix workflows, a command-line interface and integration with existing enterprise tools like Jira and Azure [4].

Cursor has gained attention because it places model interaction directly inside the editing loop, allowing codebase-aware changes, refactoring and explanation without constant context switching. Replit Agent, now in its fourth generation, extends the concept towards end-to-end application creation in a hosted environment [5]. A parallel body plan has become at least as important: terminal-native agents such as Anthropic's Claude Code and OpenAI's Codex, which live in the command line, run for extended periods, and can be wired into continuous integration.

A 2026 JetBrains survey of more than 15,000 professional developers found that 90% were using AI coding agents at work at least weekly, with 68% using them daily. Claude Code was the most widely adopted individual coding tool in the survey. [13].

The most remarkable aspect however, is the tools' (and I will use the word again) *seemingly* intelligent approach to fulfilling tasks. When given a project, Cursor will plan and logically break the problem down into manageable blocks and workflow and apply good engineering practices in designing the solutions and dealing with aspects that the user may not have thought of originally. The tool will then build the code, write tests and check results. When problems are reported it will *seemingly* intelligently hypothesise about possible causes, test these theories through ad-hoc code construction to arrive at the cause of issues and will then readily fix the problems if instructed.

"If you take the time to describe what you want (and don't want) in detail then there will be less 'schenaniganeering' as I might call it required at the end"

The process is extremely collaborative and iterative. Rarely will the initial outputs be exactly what you want, but the system will address any issues and new functionality can be easily added in a sentence or two.

In true Pareto style, you get a very long way very quickly initially and then there are what I might call "shenanigans": bending the prompts to get exactly what you want while stopping the AI undoing past successes. Also, dealing with all the new things you suddenly realised would be great to include – things you didn't know you wanted until you saw the end-to-end working executable.

In fact, how well outputs match your expectation boils down to how well you specify what you want and what you don't want (in Cursor you can define *rules* that all subsequent agents will adhere to, e.g. all code must comply with a given coding standard). If you take the time to describe what you want (and don't want) in detail then there will be less "schenaniganeering", as I might call it, required at the end.

I am finding now that the bottleneck is no longer the production of syntactically correct code, but the formulation of requirements, the detection of hallucinations, hidden assumptions, the management of dependencies, and the assurance that generated artefacts are fit for purpose. This last point is hugely important especially for safety-critical systems. I no longer need to write the code but I still need sufficient knowledge to understand the code that is generated.

From Watermill to Steam

The practical consequences are vastly accelerated productivity and the democratising of code development. I would liken it to the transition from water power to steam. Steam did not simply make mills faster; it changed where factories could be located, how labour was organised, and which businesses survived. AI in software development is having a similar systemic effect. It changes the economics of experimentation, reduces the penalty for discarding an approach, and enables small teams to attempt system-scale work previously reserved for larger organisations.

Rapidly developed prototypes are of course not certified products; acceleration at the front end does not alleviate the need for work in validation, security hardening, maintainability analysis and operational support, but the tools can help there too.



An old waterwheel: a casualty of the steam revolution. Now repurposed as a decorative landscape feature in Exmouth, Devon

In one of the workshops at the seminar in June, I asked, in the initial brief, for the IDE to write MISRA-C compliant code. I then asked it to review the generated code against MISRA-C and it found a number of violations – so as I said, it doesn't get things right first time. However, I then asked it to review the code against the specification and fix any issues, which it duly did. Similarly, security hardening if not specified at the outset will not be considered by the tool. However, once the code is written you can ask for a security review and it will report back a prioritised list of vulnerabilities, and will duly fix these if commanded.

An Evolving Tool Ecology

The current market resembles a biological radiation because many organisms are exploring adjacent niches. Some tools focus on individual programmers, others on enterprise governance, documentation, security or autonomous task completion. GitHub Copilot still benefits from deep integration with GitHub and Microsoft developer infrastructure, and now runs as an agent across editors, the GitHub workflow and the command line. Amazon Q Developer remains positioned around the Amazon Web Services ecosystem [6], while AWS has also introduced Kiro (which will replace Q Developer), a spec-first agentic environment that turns a prompt into requirements, design and tasks before code is written [14]. Google's Antigravity is an agent-first development environment, and the older Gemini CLI is being retired in its favour. Sourcegraph Cody, once a widely available assistant emphasising code search across large repositories, is now confined to enterprise deployments; the company's agentic successor, Amp, has been spun out as a separate organisation [7]. JetBrains, Atlassian and other established vendors are embedding assistants into familiar environments, and an emerging Agent Client Protocol is allowing different agents to inhabit the same editor shell.



Trilobites were not the largest Cambrian animals, but they were exceptionally well-adapted, enduring for over 270 million years

The niches continue to split. Cursor still competes by making the editor itself the centre of AI interaction, and has now launched Projects: a persistent coordinator that can direct large numbers of cloud and local agents over weeks of work, watch pull requests (request to merge code changes into a project's main repository) and continuous integration, and keep going after the laptop is closed [15]. Cognition's Devin, introduced as an AI software engineer intended to plan and execute multi-step development tasks, now also appears as Devin Desktop, the rebranded Windsurf editor, with an Agent Command Centre for supervising local and cloud agents [8]. Replit Agent 4 aims to turn natural-language ideas into deployable applications inside a shared cloud workspace. Qodo, formerly CodiumAI, has concentrated on test generation, code integrity and review assistance [9].

Poolside, Magic, Augment, OpenCode and others are pursuing large-scale coding models, open-source agents, context handling or enterprise automation. Some will become enduring platforms; others may become features inside larger ecosystems.

Survival will not be decided solely by model quality. Developers choose tools that fit habits, enterprises choose tools that satisfy governance, model suppliers choose which partners they will still serve, and safety-related organisations choose tools that leave auditable evidence. The fittest products may therefore be those that combine strong generation performance with traceability, data controls, integration into continuous integration and continuous delivery pipelines, and a credible story about intellectual property. In an ecology, the largest animal is not always the most resilient. The better adapted one usually is.

Extinction and Adaptation

The market has already produced fossils, and it is still producing them. Kite, once a prominent AI coding assistant, shut down in 2022 after concluding that the business could not be sustained despite years of technical effort [10]. DeepCode did not disappear in the same way, but was absorbed into Snyk, where its Machine Learning (ML) based code analysis became part of a broader security platform [11]. As mentioned, CodiumAI rebranded as Qodo, reflecting a shift from a narrow association with generated tests towards a wider code quality proposition. Amazon CodeWhisperer was folded into Amazon Q Developer, demonstrating how an assistant can be repositioned inside a larger enterprise AI brand and even this is now being replaced by Kiro. SpaceX's acquisition of Cursor was followed within a fortnight by OpenAI's decision to end managed access to its models inside that editor, with a proposed cut-off in November [16]. The selection pressure has not eased; it has intensified.



"These examples show that failure, acquisition and rebranding are not peripheral events. They are the selection mechanisms by which the field discovers viable forms"

These examples show that failure, acquisition and rebranding are not peripheral events. They are the selection mechanisms by which the field discovers viable forms. A tool may be technically impressive but commercially weak. Another may be less glamorous yet persist because it serves a regulated enterprise need. A tool may vanish as a product because its capability becomes a standard feature in a dominant platform, or because a supplier relationship breaks. This is exactly what one would expect during an early radiation: boundaries between species are unstable, and some organisms are transitional. Arguably, this might be classed as normal market forces, but it's the speed of these processes that is unprecedented.

The stronger candidates for enduring success currently appear to share three characteristics. First, they possess distribution through existing developer workflows, as GitHub, JetBrains and the Unix command line do. Secondly, they control or integrate with execution environments, as Replit, Amazon, cloud agent sandboxes and terminal agents do. Thirdly, they address repository-scale, persistent context rather than isolated snippets, as Cursor Projects, Claude Code and the remaining enterprise code-intelligence platforms attempt to do. None of these advantages is decisive, and today's apparent winners may still be displaced. However, these characteristics may prove important to long-term durability.

Recursive Acceleration

One consequential feature of the CambrAIn explosion is that AI code generation is being used to improve AI code generation itself and this is spanning virtually every tier of the modern AI development stack. In my experience (and for the many projects I've attempted), as long as the intent is well-specified, the tools I've used write far better and robust code than I could ever write and of course, at an incredibly fast rate. They also identify issues that I've not previously thought of, leading ultimately to higher quality code. It seems inevitable that better and more capable AI code generators will also evolve rapidly.

This recursive loop compresses the time between an underlying model improvement and its appearance in a practical development workflow. Newsletters such as TLDR AI show the resulting cadence: model releases, agent demonstrations, funding announcements, benchmark claims and product launches arrive weekly, not yearly [12]. The capability growth is already highly visible, and it has not stopped simply at larger context windows allowing reasoning over larger portions of a repository:

- Retrieval augmented generation has improved the relevance of answers by grounding them in project-specific files and documentation
- Agents can now run commands, inspect test failures and attempt repairs
- Code review assistants can suggest changes aligned with style rules
- Migration tools can transform older programming language versions or cloud service interfaces
- Terminal-native agents can run for hours, spawn sub-agents and sit inside continuous integration pipelines
- Persistent coordinators can retain project context for weeks, delegate to cloud agents, and act on signals from chat, pull requests and build failures
- Plugin and skills marketplaces let agents extend themselves, with all the supply-chain consequences that implies

This recursive acceleration also creates hazards. Benchmark-driven development can reward superficial success. Agents may learn to satisfy tests without satisfying intent. Generated code can propagate insecure "anti-patterns" at scale. Dependencies may be introduced without adequate supply-chain review. The same speed that enables exploration can multiply latent defects. Therefore, the central engineering challenge is to couple generative acceleration to verification acceleration. If the former outruns the latter, organisations will accumulate technical debt faster than they can understand it. If the understanding does not come along with the productivity, then we have a new debt: a comprehension debt. This is however not quite as bleak as it sounds as AI code generators are rather good at documenting and summarising what has been done.

There is also growing empirical literature, and it is less comforting than the marketing. A randomised trial by METR of experienced open-source developers using early-2025 tools found that they took around 19% longer when AI was allowed, even though they believed they had been sped up [17]. Later waves suggest that more agentic tools may now be accelerating some work, but the measurement is messy. My own experience remains one of dramatic speed-up on greenfield applications and workshop demonstrations. Perhaps both things can be true: new systems appear in minutes, while changes to large, high-assurance codebases can still be slowed by review, hallucination-checking and the cost of understanding what the agent just did.

From Coding to Systems Engineering

The deepest change is not that code appears faster. It is that the boundary between programming and systems engineering is moving. When an assistant can generate a database schema, application programming interface, user interface, deployment file and initial test suite from a description, the description itself becomes the critical artefact. Spec-first environments now make that explicit, writing requirements, design and task lists before they write code. Ambiguity in intent is converted into executable ambiguity. This is valuable for discovery, but dangerous where requirements are incomplete, inconsistent or safety significant.

A disciplined response is to treat AI assisted development as a transformation pipeline. Requirements, assumptions and hazards enter the pipeline; code, tests, documentation and deployment scripts leave it. The pipeline must be instrumented. Prompts, spec files, coordinator logs and plugin hashes should be preserved where they are substantively equivalent to design decisions. Model versions should be recorded. Generated changes should be reviewed against explicit acceptance criteria. Tests should not be accepted merely because the same assistant proposed them. Configuration files, infrastructure definitions and security policies deserve the same scrutiny as application code.

This viewpoint also changes education. The valuable engineer is not the person who can generate content fastest, but the person who can interrogate an output, detect a missing constraint, design a meaningful test, and recognise when a plausible explanation is wrong. Reading code becomes at least as important as writing it. Architectural judgement becomes more important, not less, because low-level implementation cost no longer dictates design choices as strongly as it once did. In short, AI is a "signal amplifier".

- Bring clear intent, strong architecture, disciplined review and it will help immensely
- Bring vague requirements, weak standards and trust without verification then disaster looms

Implications for Safety-Critical Practice

AI generated code can be used in safety-related developments today if it is subjected to the same, or stronger, assurance activities as human-produced code. Standards such as DO-178C in aviation and IEC 61508 for functional safety do not grant code moral status because a human typed it; they require evidence that appropriate processes, verification and independence have been applied. Some practitioners, however, read IEC 61508-3 as actively discouraging AI in the safety path, and current language models sit poorly with classical tool qualification: they are statistical, prompt-sensitive and not consistently reproducible. Tool qualification guidance, including the RTCA/DO-330 framework associated with RTCA/DO-178C, is also relevant when a tool could introduce or fail to detect errors, but qualifying an agent that writes, runs and patches code is a harder problem than qualifying a compiler.

The practical rule is straightforward: generated code is untrusted until justified. Human review remains essential, particularly for requirements interpretation, hazard controls and architectural decisions. Static analysis, formal methods where applicable, unit tests, integration tests, requirements traceability, coverage analysis and independent verification should be strengthened rather than relaxed.

Safety cases should explicitly state where AI was used, what controls were applied, and why residual risk is acceptable. The use of AI to generate part, or all, of the Safety case also needs to be addressed to avoid a rabbit hole of circular arguments. Organisations should also consider data governance, because sending proprietary or safety-sensitive code to external services may create confidentiality and configuration management concerns. Cloud agents, terminal agents and coordinators that watch workplace communication and team collaboration platforms or continuous integration widen that exposure. Reproducibility becomes a key consideration: will the same prompt generate the same or different code next week?

Perhaps the biggest challenges are loss of skills and "review fatigue": who wants to spend the rest of their career reviewing AI outputs?

Human factors considerations also need to be part of the mix. There are risks of automation bias, over-trust and complacency. Perhaps the biggest challenges are loss of skills and "review fatigue": who wants to spend the rest of their career reviewing AI outputs? (A theme further explored by Ben Fulford in his article).

There is, however, a developing practice in which checking is performed by a council of AI systems. Pull-request review bots, competing critic agents and multi-model checking are already shipping. Different models, tools and analysis engines can criticise one another's outputs, search for counterexamples, generate adversarial tests, compare requirements interpretations and produce evidence for human assessors. Such councils may eventually exceed individual human reviewers in breadth, patience and pattern recognition. Even then, safety responsibility will not disappear. It will move towards designing the council, validating its independence, understanding correlated failure modes, and deciding when its evidence is sufficient. Multiple AI reviewers do not necessarily provide independent assurance. Models trained on similar datasets or using similar architectures may exhibit correlated failure modes, and so, apparent agreement should not be mistaken for independent evidence.

The tools themselves have also become part of the attack surface. In September 2026 researchers disclosed a cluster of separate agent-hijacking flaws, including GitSpawn, in which a repository's local Git configuration could take over a coding agent, and Plugin4Shell, a zero-click remote code execution chain that bypassed plugin SHA-hashing in agent marketplaces [18].

Some vendors patched the affected implementations quickly. Other affected products were already being retired or had not yet received a corresponding client-side fix. For a safety organisation this is not a curious software-supply-chain anecdote. The coding agent now sits inside the trusted computing base: it can read the repository, run commands, install dependencies and, in some configurations, act on external signals without a human in the loop. Permitted-tool policies therefore need to cover plugin sources, auto-update, Model Context Protocol (MCP) servers[1], and whether a cloud agent is allowed to watch chat channels or build pipelines.

[1] An MCP is a standard-based bridge that mediates and controls access between AI systems and your local filesystems, databases, enterprise tools (like GitHub, Jira, or Slack), and internal enterprise APIs.

Conclusions

The Cambrian Evolutionary Explosion did not produce a finished animal kingdom overnight. It produced an extraordinary burst of experimentation: new forms appeared, ecological niches opened, competition intensified, and many evolutionary experiments disappeared while others succeeded. The resulting ecosystem was fundamentally different.

The CambrAIn Explosion may prove similar. AI-assisted development is moving from tools that help programmers write code towards agents that can interpret specifications, plan work, create and modify software, run tests and operate for long periods. This isn't just about making software faster – it is changing how development is structured and funded. As a result, small teams can now build and maintain complex systems that used to require massive engineering budgets. But speed of generation is not the same as speed of assurance. For safety-critical engineering, the central challenge is whether our ability to understand, verify, validate and govern what is generated can keep pace with accelerating production.

This changes the role of the engineer. Skills include specifying intent, identifying hazards and hidden assumptions, challenging plausible, incorrect outputs, designing verification and determining when evidence is sufficient. AI may reduce the amount of code engineers write, but it does not reduce their responsibility for the systems that code creates.

AI may also provide new assurance mechanisms: automated analysis, adversarial testing, competing verification agents, formal reasoning and continuous evidence generation. But these mechanisms must themselves be understood and controlled. Multiple AI systems agreeing with one another do not necessarily provide independent evidence. Our objective should not be to stop the CambrAIn Explosion, nor to accept it uncritically. It should be to ensure that assurance evolves at least as quickly as generation. The opportunity is enormous. So is the responsibility.

What should SCSC members do?

The SCSC community should engage with the CambrAIn Explosion early, critically and practically. Waiting for the market to stabilise may feel prudent, but it risks leaving assurance practice several years behind development practice. Conversely, adopting tools without disciplined governance would confuse productivity with dependability. The correct suggested approach is experimental conservatism: learn quickly, deploy carefully, and collect evidence. Organisations should run pilots on non-critical codebases, measure defect patterns, compare review effort, and identify which tasks are genuinely accelerated. They should also update development standards so that AI use is visible rather than informal. Three actions seem particularly urgent:

- Safety teams should create policies for permitted tools, data handling, prompt retention, model version recording, plugin sources, auto-update, and the use of cloud or terminal agents on safety-related repositories
- Assurance teams should develop review checklists for generated artefacts, including tests, documentation, spec files and coordinator logs
- Professional bodies should share incident reports, near misses and good practice patterns involving AI assisted development. Studies such as METR's are a start, but the field still needs empirical evidence from safety-related work, not anecdotes alone

References

[1] Wikipedia contributors, "Cambrian explosion", https://en.wikipedia.org/wiki/Cambrian_explosion, accessed September 2026.

[2] SCSC Seminar: How to Make the Most of AI: A Workshop for Safety Engineers, 25th June 2026, scsc.uk/e5101

[3] Anysphere, "Cursor: The AI Code Editor", https://www.cursor.com, accessed September 2026.

[4] GitHub, "GitHub Copilot", https://github.com/features/copilot, accessed September 2026.

[5] Replit, "Introducing Replit Agent 4: Built for Creativity", https://replit.com/blog/introducing-agent-4-built-for-creativity, accessed September 2026.

[6] Amazon Web Services, "Amazon Q Developer", https://aws.amazon.com/q/developer, accessed September 2026.

[7] Sourcegraph, "Cody FAQs", https://sourcegraph.com/docs/cody/faq, accessed September 2026.

[8] Cognition, "Introducing Devin Desktop", https://cognition.com/blog/introducing-devin-desktop, accessed September 2026.

[9] Qodo, "CodiumAI is now Qodo", https://www.qodo.ai/blog/introducing-qodo-a-new-name-the-same-commitment-to-quality, accessed September 2026.

[10] Kite, "Kite is saying farewell", https://web.archive.org/web/20221117130927/https://www.kite.com/blog/product/kite-is-saying-farewell/, accessed September 2026.

[11] Snyk, "Snyk acquires DeepCode", https://snyk.io/news/snyk-acquires-deepcode/, accessed September 2026.

[12] TLDR, "TLDR AI archives", https://tldr.tech/ai/archives, accessed September 2026.

[13] JetBrains, "AI Coding Agents: Adoption Trends", https://blog.jetbrains.com/research/2026/08/ai-coding-agent-adoption-2026/, accessed September 2026.

[14] Kiro, "Kiro: Move beyond AI coding to agentic engineering", https://kiro.dev/, accessed September 2026.

[15] Cursor, "Introducing Projects", https://cursor.com/blog/projects, accessed September 2026.

[16] OpenAI, "Our decision on Cursor following its acquisition by SpaceX", https://openai.com/index/our-decision-on-cursor-following-its-acquisition-by-spacex/, accessed September 2026.

[17] METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity", https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/, accessed September 2026.

[18] Lyons, J., "AI coding agents' 0-click RCE flaw could hand attackers keys to the kingdom", The Register, 17 September 2026, https://www.theregister.com/security/2026/09/17/ai-coding-agents-0-click-rce-flaw-could-hand-attackers-keys-to-the-kingdom/5297335, accessed September 2026.

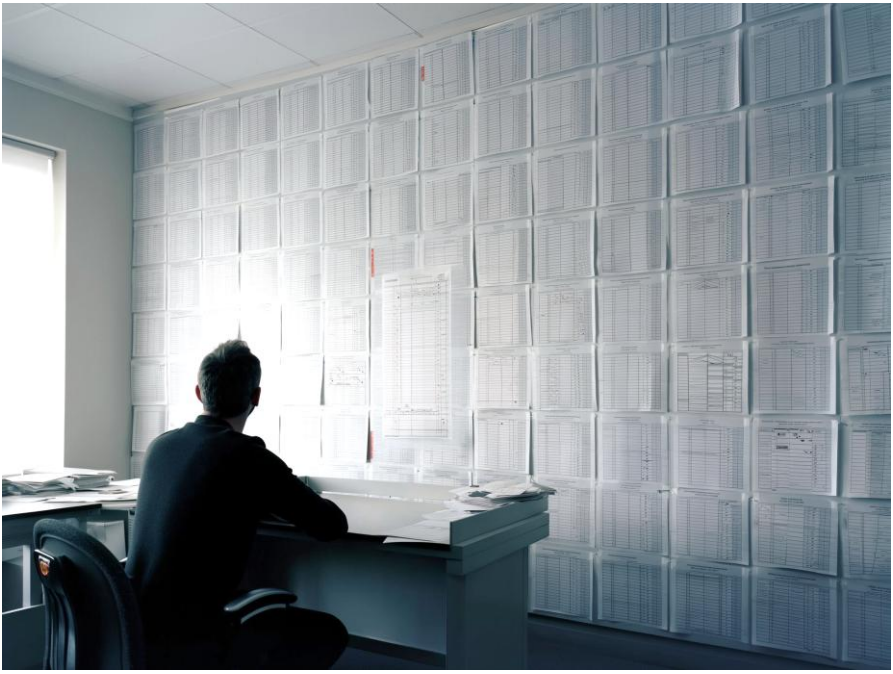
Image attribution:
top image: AI-generated (Gemini), The Cursor code editor logo: Courtesy of Anysphere, Inc., trilobite fossil: ID 75353597 | Cambrian © Eugen Thome | Dreamstime.com, Snyk and DeepCode integration graphic: Courtesy of Snyk Ltd.

Paul Hampton, Timegalore Ltd.



Paul is a Chartered Engineer with over 35 years' experience in IT. He has spent 20 of those designing and developing enterprise systems in sectors such as Energy & Utilities, Government, Criminal Justice and Health and 15 years in System Safety in a variety of capacities, including safety engineering, safety management, independent auditing and corporate governance and assurance. He has published a number of articles and papers and is a frequent presenter at industry seminars and symposiums. In 2019 Paul joined the SCSC Steering Group and Executive Committee and is the SCSC Newsletter Editor and Web Master.

I don't want to spend the rest of my career reviewing AI outputs



Arguably, generative Artificial Intelligence (AI) can be added to safety flows as long as we keep humans in the loop – humans can check the AI outputs as one might the output of a junior engineer. However, AI can vastly accelerate output generation and simply overwhelm reviewers with data. How, therefore, can we assure the AI outputs, especially when tool quality increases and errors become rarer? Ben Fulford discusses these issues and presents an approach to filtering the data to focus on what's important from a safety engineering perspective.

I used to think the answer to putting AI into safety workflows was simple enough. Keep humans firmly in control of every output. Treat what the model produces the way you would treat work from a junior engineer: read it, question it, check the reasoning, sign it off. Do that thoroughly and the risk is managed.

Over the last few months I have stopped believing that is feasible.

I have been building a GraphRAG (Graph-based Retrieval-Augmented Generation) tool for hazard identification. It reasons over a structured model of the system and produces hazard analysis that is accurate and well-evidenced, and it produces a great deal of it. Thousands of lines of technical content in a few minutes.

The first time I sat down in front of a full run, I could not work out where to start. The quality was there. The volume defeated me.

The problem the tool was built to solve is a real one. Hazard analysis is slow and expensive. It occupies experienced engineers for days at a time, it usually gets done once, and the amount a programme can afford is set by how many engineer-days can be found rather than by how much analysis the system warrants. The tool removes that constraint almost completely, and I am convinced that what I have built is a real step forward in how this work gets done.

Removing that constraint created a second problem. Somebody now has to sit down and confirm that all that output is correct. The point of building it was to give engineers their time back. What I built alongside it was a new job, and it is not a job I want to do.

This is worth working through in public, because it is not specific to my tool. Any team putting generative AI into an assurance workflow will meet this within weeks of the tool starting to work properly.

A world of abundance

For most of history, information was scarce. Creating it took effort, recording it took materials, and copying it took somebody's time. That cost acted as a filter. If a thing had been written down and kept, somebody had decided it was worth the trouble, and the reader could take that decision as a signal.

Generative AI almost completely removes the cost of production. Set alongside the infinite scroll feed and the thousand-word email that used to be three lines, we now live with an abundance of information and no matching increase in our capacity to take it in. The filter that scarcity provided has gone, and nothing has replaced it. Herbert Simon saw this coming in 1971 [1]:

"We now live with an abundance of information and no matching increase in our capacity to take it in"

> *"In an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention."*

Attention is now the scarce resource. The people who can hold theirs on the thing that matters, in an environment engineered to fragment it, have an advantage over everyone else, and that advantage is growing.

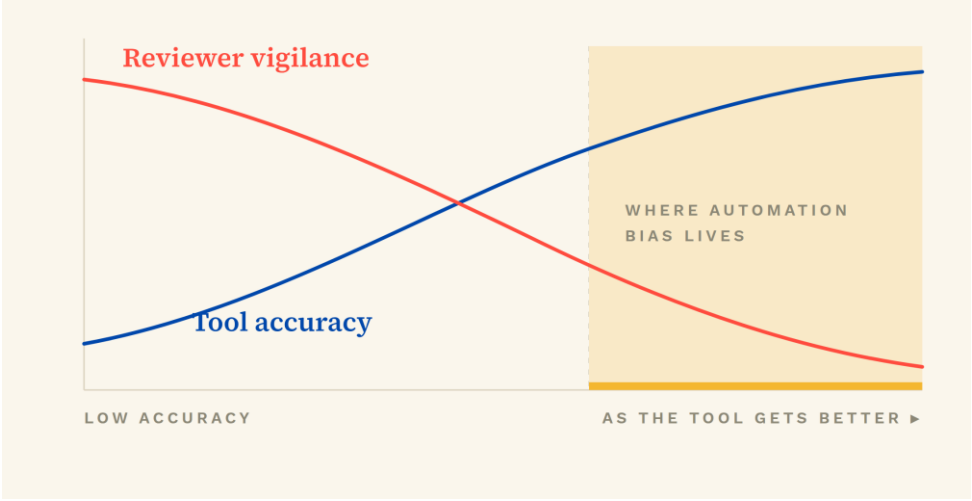
The same logic applies to engineering work, with one difference: we get to design the system rather than rely on individual discipline. Safety engineering has always been an exercise in directing limited attention. A hazard log entry cost somebody an afternoon of thought, so entries were deliberate. A safety case was long because somebody spent months on it. Scarcity did a lot of that filtering for us, without anybody designing it that way. With generation now close to free, the filtering must be designed deliberately, and I think the safety engineering community is better placed than most to design it.

Why reading everything stops working

Reading everything fails for two reasons, and the second is worse than the first.

The first is arithmetic. A hazard and operability study (HAZOP) team working through a two-day workshop might produce a few hundred worksheet rows. The same coverage now comes back in an afternoon, at greater depth, with the option of running Failure Modes and Effects Analysis (FMEA), Fault Tree Analysis (FTA) and System-Theoretic Process Analysis (STPA) over the same system model on the same day. The volume of analysis went up by an order of magnitude. Review capacity did not move. Adding reviewers is the obvious answer, and it is a poor one, because the cost curve is linear in exactly the dimension the tool made cheap.

The second reason is human factors. At low accuracy, errors are uncomfortable and visible. The failure rate is high enough to feel, so engineers stay alert. As the tool improves, trust rises and checking drops. Automation bias sets in; the tendency to accept an output because outputs are usually right. The errors that survive this are the ones that most resemble correct answers, which makes them hardest to catch at precisely the point people are looking least carefully.



Lisanne Bainbridge described the mechanism in 1983 [2], forty years before a Large Language Model (LLM) walked into it. Ironies of Automation is four pages long and every paragraph of it applies here. Three lines in particular:

> *"It is impossible for even a highly motivated human being to maintain effective visual attention towards a source of information on which very little happens, for more than about half an hour."*

> *"Physical skills deteriorate when they are not used, particularly the refinements of gain and timing."*

> *"By taking away the easy parts of his task, automation can make the difficult parts of the human operator's task more difficult."*

Every control-room lesson in that paper maps onto an engineer working through several hundred rows of AI-generated analysis. Human in the loop, implemented as a person reading everything, is a control that degrades as the tool gets better. That is worth stating plainly, because it is the control most organisations are currently relying on, and the one most likely to turn up unexamined in a safety argument.

"Human in the loop, implemented as a person reading everything, is a control that degrades as the tool gets better"

There is a third cost, less dramatic and more corrosive. Reading everything spends your most experienced engineers on the part of the work that needs them least.

What software engineering already did

Software engineering worked through a version of this problem, and the answer is instructive.

A compiler converts source code into machine instructions. Very few review the machine instructions. That was not always obviously safe, and early compilers were distrusted for good reason. Trust was earned by building apparatus around the tool rather than by checking its output line by line, in three steps.

- One: constrain what the tool can produce. Type systems, language rules and a defined specification of valid output mean whole classes of error cannot be expressed in the first place.
- Two: verify the output against that specification automatically. Test suites check behaviour against what the code was supposed to do. The check is machine-run, repeatable and cheap enough to run on every change.
- Three: monitor in service and make reversal cheap. Production monitoring catches the errors that got through and the drift that builds up over time, and rollback is a decision rather than a project.

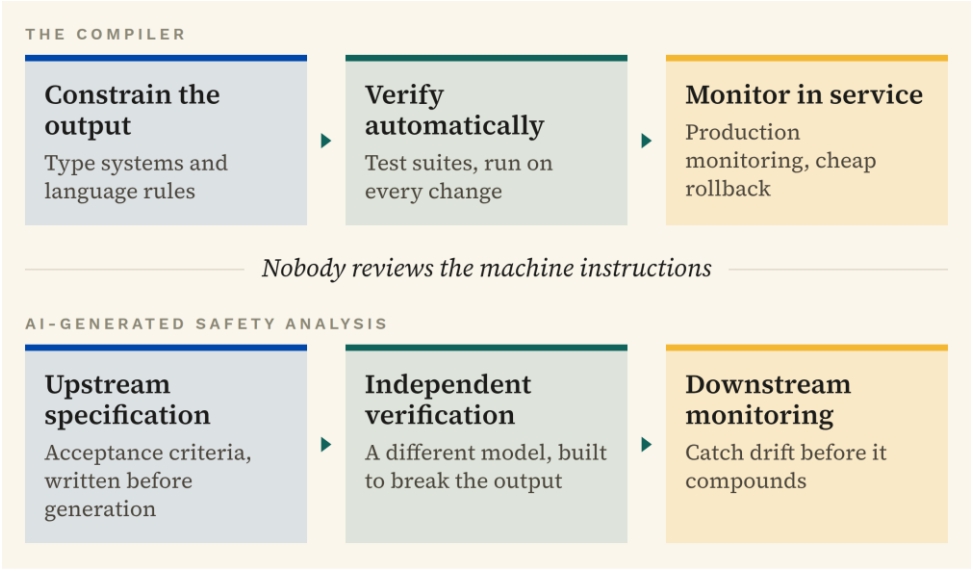
Tool qualification under IEC 61508 [3] and RTCA/DO-178C [4] formalises the argument. The manual middle was removed, and the profession is more confident in compiled output today than it was when people were checking it by hand.

That structure maps onto safety assessment directly.

Upstream specification. Define what a good output has to satisfy before generation starts, precisely enough to test against. The standard in play, the risk appetite, the acceptance criteria, the scope boundary. Most teams do this after the fact, inside a reviewer's head, one output at a time. Writing it down first is what makes everything downstream possible.

Independent verification. A second model, ideally a different architecture, built specifically to try to break the output against that specification. This is the only layer that scales at the volume these tools produce. It is also the layer people are most sceptical about, and the scepticism is well founded: the checker has failure modes of its own, and they correlate with the generator's if they are careless about diversity. Common-cause failure is a live concern here, and it is one this field already knows how to reason about.

Downstream monitoring. Track how the review process performs over time, with enough granularity to catch drift before it compounds. If a class of finding is being accepted and later found wrong, that is a process capability problem and it should surface as one.



One result from my own build is worth reporting here, because it bears directly on where the effort should go. On the GraphRAG tool, the guardrails, validators and self-checks I put around the model improved output quality more than swapping in a more capable model did. Engineering the system around the AI mattered more than the AI. I did not expect the gap to be as large as it was, and it is the single finding that most changed how I am spending development time.

Design the output to be cheap to check

There is a fourth move, and it changes the economics more than any of the above.

Verifying a claim costs far less than generating it, provided the output was built for verification. A finding that arrives as a paragraph of prose has to be re-derived before anyone can check it. The same finding, arriving as a typed claim marked as fact, inference or assumption, carrying a citation into the system model it came from, can be checked by following the citation. Identical content, an order of magnitude difference in review cost.

The tool I am building carries a claims ledger for this reason. Every claim has a status and a provenance link. Review burden falls with no change to the underlying model.

This argument is usually made from trust and regulation. The version engineering managers act on is the labour economics one.

Consequence and confidence

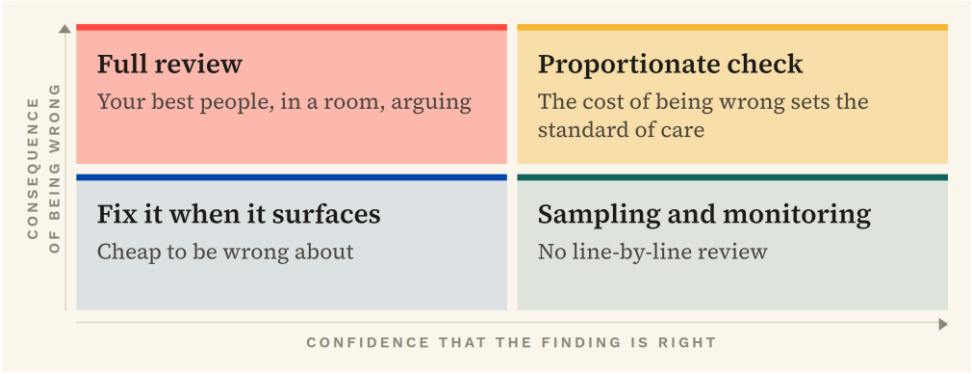
Apparatus reduces the volume that needs a human. It does not decide where the humans go. That decision needs two variables: the consequence of a finding being wrong, and the confidence that it is right.

High consequence, low confidence is where human attention earns its cost. These get full review, and they get your best people. Pull together a mixed group: the domain specialist, the operator, the person who worked the last incident, the resident sceptic. This is the small population of items where a room full of expensive expertise is the correct answer.

Low consequence, high confidence needs no line-by-line review at all. Sampling and monitoring are adequate, and the argument for why they are adequate can be written down and defended.

High consequence, high confidence still earns a proportionate check, because the cost of being wrong is what sets the standard of care.

Low consequence, low confidence is cheap to be wrong about. Fix it when it surfaces.



None of this should feel unfamiliar. It is what safety engineering has always done. Every risk is ranked by severity and likelihood before anybody decides how much attention it deserves. As Low As Reasonably Practicable (ALARP) is, in the end, an argument about where to stop spending effort. Manufacturing solved "we cannot inspect everything" in the 1920s with acceptance sampling, and the resulting framework, later MIL-STD-105 [5] and ISO 2859 [6], gives a written, auditable argument for partial inspection with quantified producer's and consumer's risk. Regulators and quality departments have had a century to object to it.

We have never claimed to examine everything with equal care. We have claimed to be able to explain where we looked hardest and why. That is the discipline this problem needs, and it is why the safety engineering community is better placed to lead this work than most AI teams. Ranking by consequence before deciding what deserves scrutiny is our default setting.

Angus Keddie put the point to me more sharply than I had managed. He is comfortable with the tool hallucinating, provided it reliably catches the high-consequence issues. He calls those HAZOP gems: the specific, difficult edge cases that lead to a serious outcome. The team's job is to argue about those. Sometimes the argument ends with the team dismissing the flagged issue as not credible, and that is still time well spent, because it was spent where being wrong costs the most.

I would add one thing to that. The gems are the interesting part of the job.

The part I have not solved

All this rests on a confidence score you can trust, and that is the part I have not solved.

A model's self-reported confidence is a generated token like any other. If the number directing engineer attention is itself uncalibrated model output, the human-in-the-loop control inherits the exact weakness it exists to manage.

What the apparatus needs is an independent judge, on the same principle as the compiler: the check sits outside the thing being checked, it runs against a stated specification, and its own performance can be measured over time. A judge that shares the generator's architecture, its context and its blind spots is not an independent check at all.

There is a second requirement that gets discussed less. The judge must be asking the right question. A score that measures whether a finding is well formed, internally consistent or plausible will point engineers towards tidy output. The question that matters is whether the finding is correct, and whether it is correct about something that counts. Ask the wrong question and the score confidently directs attention to the wrong place, which is the failure the whole system exists to prevent.

Building that judge is the hardest problem in the stack. It is the subject of my next article.

What the job becomes

Here is the version of the work I want, and the one I think is achievable.

The tool generates the analysis. The apparatus around it, specification, independent verification and monitoring, absorbs the volume that no human should be reading. What comes out arrives with clear markers on it, showing engineers where their effort is needed and why it is needed there.

The engineers go where the markers point. The difficult cases, where being wrong carries a real consequence and the system cannot vouch for its own answer, get a room and an argument. Some of those end with the team dismissing the finding as not credible, which is a good use of the time. Some of them change the design.

That is a better day than the one I had staring at a full run with no idea where to begin. It is also, by some distance, the most interesting part of this profession. The judgement calls at the edges, argued out by people who have seen things go wrong, are why most of us are still doing this work.

The risk is that we get the other version by default. Adopt these tools without building the apparatus and the review job arrives anyway: unfunded, undesigned, and quietly loaded onto the engineers who were promised their time back. Human in the loop becomes a line in a safety argument that nobody has tested. That is a worse position than not adopting the tools at all, because it carries the appearance of a control.

We already have the methods. Ranking by consequence, arguing about where to stop, writing down why partial coverage is adequate and defending it to a regulator: that is the job description. The new part is applying it to the output of a machine rather than the output of a team and doing it before the volume becomes unmanageable rather than after.

I do not want to spend the rest of my career reviewing AI outputs. I do not think I will have to. That depends on this community designing the system that means none of us do, and the time to design it is now, while the volume is still merely uncomfortable.

References

[1] H. A. Simon, "Designing Organizations for an Information-Rich World", in M. Greenberger (ed.), Computers, Communications and the Public Interest, Johns Hopkins Press, 1971, pp. 37-72

[2] L. Bainbridge, "Ironies of Automation", Automatica, Vol. 19, No. 6, 1983, pp. 775-779

[3] IEC 61508:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems, International Electrotechnical Commission, 2010

[4] RTCA DO-178C / EUROCAE ED-12C, Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc., 2011

[5] MIL-STD-105E, Sampling Procedures and Tables for Inspection by Attributes, US Department of Defense, 1989 (cancelled 1995)

[6] BS EN ISO 2859-1:1999, Sampling procedures for inspection by attributes, Part 1: Sampling schemes indexed by acceptance quality limit (AQL) for lot-by-lot inspection, ISO, 1999



Ben Fulford, Safety and Assurance Engineering, BMT

Ben builds tools for safety and assurance engineering: hazard identification, hazard log modernisation, and training that develops judgement rather than attendance records. He has 15 years across defence, maritime and nuclear, including safety manager on the Astute submarine platform management system and lead independent safety auditor on an uncrewed underwater vehicle programme. He is currently building a hazard identification tool on a knowledge graph, and a browser game that teaches AI assurance through consequence.

The author retains copyright of this article.

Devils in the Detail



In this, the 4th and final episode of a series of articles exploring the principles of ontological modelling, the Ontology Working Group (OWG) builds on previous episodes, introducing some new terms from the Unified Foundation Ontology (UFO) and expanding further on the worked example relating to flights and flight planning.

With these articles, the OWG aims to make the topic more accessible through a series of connected newsletter articles. Each episode expands on ontological aspects and, by the end, will hopefully provide the reader with enough knowledge to read and contribute to models in the future.

What Is an Ontology?

As we mentioned in earlier articles, an ontology is not some esoteric formalisation, which is nice to have, but is essential to people's day-to-day work on real systems.

When we build a system, we are modelling our world and this involves developing concepts, their relationships and the rules for connecting them – this is called an ontology, or more formally: "**an explicit specification of a conceptualization**" [1].

If not explicitly specified, establishing a shared conceptualisation, that is an understanding of interrelated concepts, will be much harder and miscommunication may be more common. Such shared understanding helps ensure that models used to support parts of the lifecycle of a system always remain coherent and consistent. If the concepts in the model are not defined in writing, meaning can just be held in the minds of the designers and be lost when they are no longer involved. An ontology serves as a contract of meaning between stakeholders, who make what are known as 'ontological commitments'. That is, they commit to produce engineering artefacts in a way that is consistent with the ontology that captures their shared conceptualization.

Where Did We Get To? – A Recap

We are now on our 4th episode (1st in Feb 2025, 2nd in Oct 2025 and 3rd in Feb 2026). This article should still be of interest if you have not read the previous three, but you will get a lot more value out of it if you read them first.

To recap on the earlier articles, the OWG uses the **Unified Foundation Ontology (UFO)** [2]. This is a foundation for ontological modelling implemented in **OntoUML** [3] (a customisation of the Unified Modelling Language, UML [4], with associated software support via application plugins [5]). The following notation was previously introduced.

- We use boxes and connecting lines (more formally **Class**es and **Relationship**s using the modelling terminology, which are highlighted in this section in **bold**) to represent concepts in our worldview. For readability, specific **Class**es mentioned in the text of this article are shown in **bold purple**.
- Ontological aspects are expressed as **Stereotypes** (the part of the **Class** definition with Guillemets like this: **«role»**). **Stereotypes** mentioned in the text in this article are in **bold blue**.
- Arrows with a closed, unfilled triangle —▷ are **Generalizations**, or in other words, one 'thing' subtypes the more general 'thing' that is being pointed to (for example, a **Cessna** is a subtype of **Aircraft** and **Aircraft** is a **Generalization** of **Cessna**)
- An open arrow —> shows an **Association** between things. For example, in the phrase *"Pilot 'flies' Aircraft"*, the concept 'flies' would be modelled as an **Association**. Note that **Associations** also have **Stereotypes**.
- **Association ends** carry numerical expressions at the end of each association, such as **0..*** and **1..*** (read as *zero or more* and *one or more* respectively). These show **Multiplicity** – the valid range of the number of each type of thing that can be related. This is used to help express constraints, such as a **Pilot** can be trained to fly one or more **Aircraft**, but a given **Aircraft** can usually be flown by, at most, two **Pilot**s at any one time.

We covered the important concept of identity principles: these form sets of characteristics that, in combination, give us the ability to separate one individual entity from another (individuate, more formally).

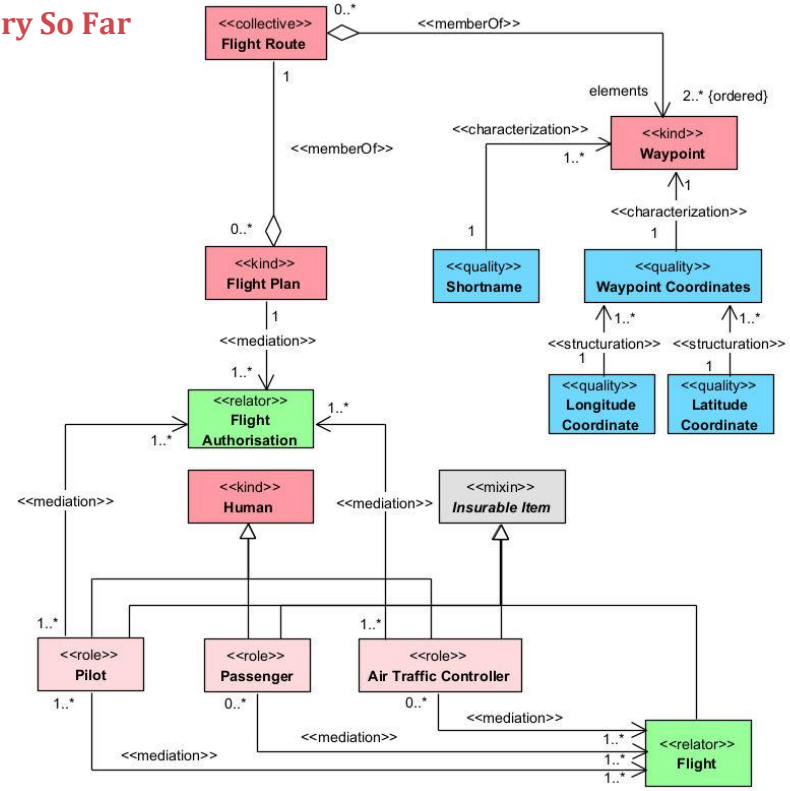
We have also covered some core UFO concepts, expressed as **Stereotypes**:

- **Kind**s reflect the objects we find around us like people, aircraft and dogs that share a common identity principle. **Subkind**s allow us to build classic taxonomies.
- **Role**s and **Phase**s are types whose instances may be only temporary while the thing to which they applied still exists. The same **Human** may hold the **Role**s **Pilot** and **Passenger** and go through **Phases** of **Employed** and **Unemployed** at different times during their lifetime. We will explore in this article how logical rules may be needed to prevent conflicts in models that use these concepts.
- **Mixin**s are used to describe shared properties of objects that do not all belong to the same **Type** and so can span multiple **Type**s with differing identity principles.

In the last edition we also introduced the higher order categorisations of **Sortals**: things we can individuate and count (like **Pilot**s and **Waypoint**s) and things we can't, like **Loud**, **Metal** and **Fatigue** which are **NonSortals**. We also discussed rigidity: **Rigid** **Class**es like an **Aircraft** which is an **Aircraft** throughout its entire life as opposed to **AntiRigid** like an **Airborne Aircraft**, which exists only for part of the **Aircraft**'s lifetime.

We finally expanded on an example based on **Flight Plan**s and **Waypoint**s to introduce the idea of a **Collective** of **Waypoints** forming the **Flight Route** and how **Quality** can be used with the stereotyped **Association** **Characterization** to describe what individuates a **Waypoint**. We also introduced **Structuration** to decompose different aspects of a **Quality**

Our Story So Far



We've developed our story over the last three episodes to arrive at the previous figure.

In the diagram, a **Flight Route** is a **Collective** associated with a number of **Waypoint**s which are representations of physical locations on the Earth the **Pilot** intends to navigate between.

Waypoints have exactly one **Waypoint Coordinates**, which is modelled as **Latitude Coordinate** and **Longitude Coordinate**. Coordinates are alphanumeric values representing physical coordinate points on the Earth. For example, for the **Waypoint** with **Shortname** **ETOXU**, the **Latitude Coordinate** might have the representation: **N47 43.6** and **Longitude Coordinate:** **E009 33.0**

At the end of the last episode, we left the reader to consider a couple of questions that explore how robust our model is at the moment:

- In reality, an individual **Human** cannot be both a **Pilot** and a **Passenger** on the same **Flight** – does the model prohibit this situation?
- Does the model prohibit **Waypoints** located at different **Waypoint Coordinates** from having the same **Shortname**? If not, do you see any problems that might arise?

Let's explore the answers to these questions next.

A Crisis of Identity

The lower part of our model shows that **Pilot**, **Passenger** and **Air Traffic Controller** are roles of **Human** and shows their relationships to a **Flight**. The **Multiplicity** tells us a great deal about the relationships. A **Flight** must have a least one **Pilot** (1..*), it does not always need to have an **Air Traffic Controller** (0..*), although it might have several, and it does not necessarily need to have a **Passenger** (0..*), although it might have many. It also does not prevent a **Pilot** from also being a **Passenger** on the same **Flight**; so the model is under-constrained; it is modelling things as being independent when there are in fact relationships between them not reflected in the model.

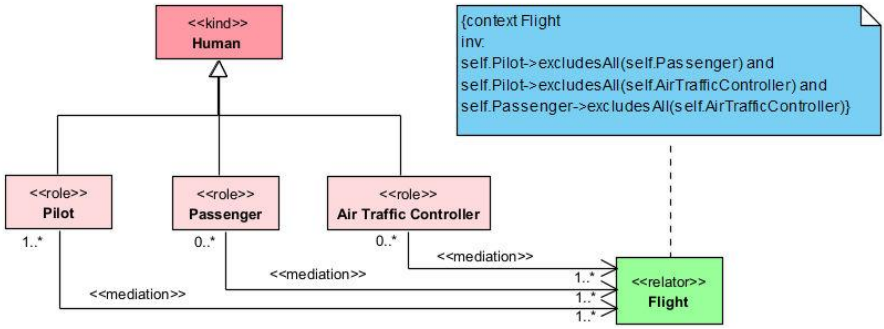
This is an example of a modelling pattern that designers may gravitate towards but in fact has pitfalls and so is classed as an *anti-pattern*. In this particular case, the "anti-pattern" is called the "Relator Mediating Overlapping Types" (or RelOver for short) [6]. (This is just one of many anti-patterns; see [9] for more examples).

This illustrates the wonderful power of ontologies; the defined **Class**es, relationships and rules ensure meaning can be stated in a way that helps reduce bias, preconceptions and implicit assumptions. In this case we see that while a model may be well-formed according to the language used to create the model, it may not precisely model the real-world situation you intended!

"This illustrates the wonderful power of ontologies; through their preciseness; the model defines the classes and their relationships so meaning can be stated in a way that helps reduce bias, preconceptions and implicit assumptions"

Clearly, we want our model to prevent illogical combinations, so we can add a constraint to indicate this in the model.

Many languages can be used to express constraints, the one we will explore is packaged within the UML specification and is called the Object Constraint Language (OCL). This has its own syntax, which we won't have time to explore in full in this article, but suffice to say it allows for more code-like specification of custom constraints. See the following diagram (other model elements have been omitted for clarity).



In this diagram we have added an OCL constraint on the **Flight** relator:

```
context Flight
inv:
self.Pilot->excludesAll(self.Passenger) and
self.Pilot->excludesAll(self.AirTrafficController) and
self.Passenger->excludesAll(self.AirTrafficController)
```

The notation `self.` refers here to the current instance of **Flight** being considered when the constraint is evaluated. For example, the real **Human**s that took the **Flight** denoted by "French Bee flight BF731 between Los Angeles and Paris Orly on the 28th Dec 2023". Note that it is the set of **Human**s because that is common **Kind** shared by all of the **Role**s that are subject to the constraint (indicated by the **Generalization**).

`excludesAll(…)` is an operation that ensures, within the set of real **Human**s associated with a given **Flight**, that no single individual can simultaneously belong to both the source set (to left of the `->`) and the target set (inside the `(…)`).

The `and` operator serves its usual function in logic.

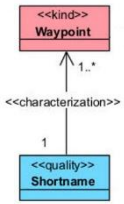
So the above constraint code is saying that, in the `context` of a **Flight**, we have an invariant (`inv`) constraint that ensures, for any given instance of **Flight** that:

- A **Pilot** cannot also be a **Passenger** on the same **Flight**
- A **Pilot** cannot be an **Air Traffic Controller** on the same **Flight**
- A **Passenger** cannot be an **Air Traffic Controller** on the same **Flight**

Because the rules apply in both directions, the three elements of the constraint are sufficient to cover all possible pairings of **Pilot**, **Air Traffic Controller** and **Passenger**.

Note that the constraint only applies to instances of each of the concepts named in the constraint. In this case all the concepts have the type **Role**. Since the roles **Pilot**, **Air Traffic Controller** and **Passenger** only apply to a specific real-world instance of **Flight**, this means that the same **Human** may at different times travel on a **Flight** in each of these roles. **Pilot**s quite like to fly as **Passenger**s when they go on holiday, for example!

Losing Our Way(point)



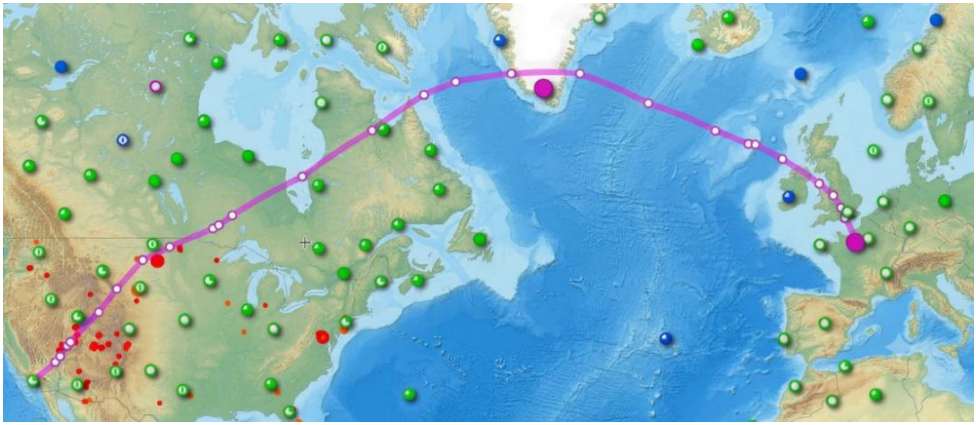
The second question we asked was whether the model supported different **Waypoint**s with the same **Shortname**. The multiplicity between **Waypoint** and **Shortname** tells us the story. While a **Waypoint** must have one and only one **Shortname**, a particular **Shortname** can be related to more than one **Waypoint** even though they may have different **Waypoint Coordinates**.

As you might guess, allowing duplicate **Waypoint Shortname**s could, and has been, problematic. In particular we will look at one such incident: the NATS Air Traffic Control (ATC) outage that occurred on the 28th Dec 2023. A duplicate **Waypoint Shortname** (**DVL** in this instance) caused a system outage that led to 700,000 passengers suffering cancellations and a cost to the industry estimated to be between £75m and £100m [7].

NATS reported that the issue occurred after 15,000,000 successful flights, so what combination of events led to the problem?

To explain the problem, we need to understand how flight planning works for international flights that might span several airspace jurisdictions, such as US, UK and French airspace. This explanation is based on one of the CAA reports of the incident [8] (CAP2993C).

The flight in question was a French Bee flight BF731, which routinely flies between Los Angeles and Paris Orly every two or three days. Normally the flight plan would be more direct across the Atlantic thus missing UK airspace entirely (this would have financial benefits) but occasionally, weather conditions dictate a more eccentric northerly route which will necessarily mean passing into UK airspace. Here is the original planned route:



As the route is unusually more northerly, it takes the flight plan, through North Dakota and hence one of the problem **Waypoint**s at Devil's Lake (DVL).

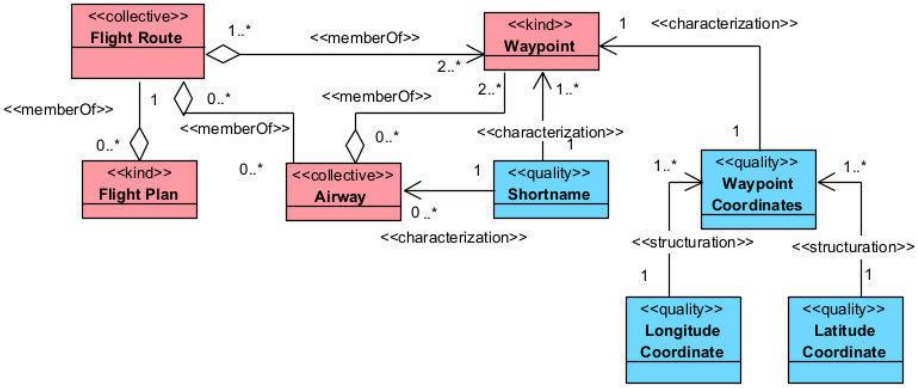
The flight plan as a collection of **Waypoint**s (read from left to right) is as follows:

```
KLAX ORCKA5 LAS Q70 BLIPP Q842 WINEN MLF J107 OCS CZI DIK DVL VESRU
YXL ELVEL PIGLA 5870N MIBNO MAXAR 6250N 6240N 6030N 5820N SUNOT KESIX
APSOV SOSIM L15 KEPAD L151 KIDLI UN859 GWC UN859 LGL A34 BOBSA LFPO
```

KLAX and LFPO at the beginning and end of the sequence are the airport codes for Los Angeles and Paris Orly. Let's call this, informally, the "FB plan" for French Bee flight plan.

Note the **Waypoint Shortname**s of the form: "6030N"; these are when the aircraft travel over Atlantic regions where there are no ground-based radar or radio navigation beacons to name points after. The **Shortname** therefore takes a shortened abbreviation of the GPS position. This still fits with our model, but there is something else: look at **Waypoint**s called **L15**, **L151** and **UN859**. These look like normal **Waypoint** names but they are actually **Airway**s that are in themselves, a **Collective** of **Waypoint**s.

These particular corridors are used so frequently along with the associated **Waypoint**s that they have been summarised as a **Collective** with its own **Shortname**. Once on an **Airway**, the aircraft's flight management system will typically track through the intermediate **Waypoint**s automatically. We need to adjust our model to support that possibility as follows:

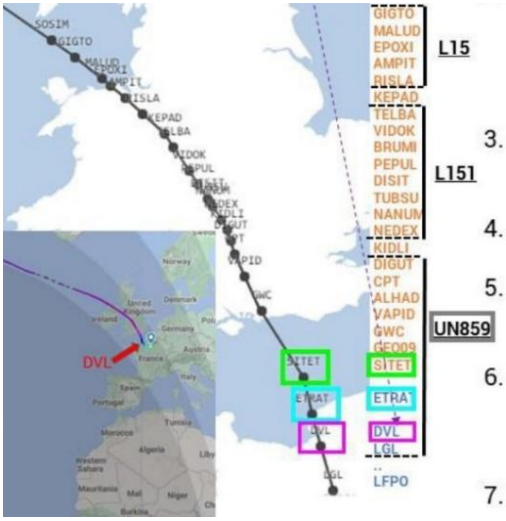


So we now have a new **Class** called **Airway**, which is related to **Flight Route**, **Waypoint** and **Shortname**. The multiplicity tells us the following:

- A **Flight Route** *may* contain zero or several (note the 0..* multiplicity) **Airway**s and an **Airway** *may* be included in one or more **Flight Route**s
- An **Airway** has one and only one **Shortname** and a **Shortname** may refer to one or more **Airway**s. (Yes, **Airway Shortname**s are not globally unique either)
- An **Airway** has two or more **Waypoint**s and a **Waypoint** may be included in one or more **Airway**s

Returning to the NATS problem, you can see the **Waypoint** with **Shortname** **DVL** in the FB plan equating to the Devil's Lake beacon in the US.

For flights crossing European airspace, the flight plan must be submitted to EUROCONTROL for approval first. This organisation processes all flight plans requiring services from its member state Air Navigation Service Providers (ANSPs), such as NATS and their system (IFPS) converts the flight plans from an ICAO document 4444 (ICAO4444) format to another format known as ADEXP, which is a European-wide flight plan specification that includes, amongst other data, additional geographical **Waypoint**s within the European region (added by EUROCONTROL) necessary to the route of a flight.



A critical **Waypoint** in the list is **APSOV**. This was in the original flight plan and denotes the entry point into UK airspace.

NATS, as part of its processing, needs to determine the exit point from UK airspace and it has some logic to do this. The logic runs through candidate exit points and discards those added by EUROCONTROL in preference to one that was in the original **Flight Plan**.

"The backup system then took over and repeated the processing resulting in the same conclusion – a critical exception, thus bringing all processing to a standstill"

As the aircraft was following **Airway** UN859, the expected exit point was **SITET**, but this was not in the original **Flight Plan** so was rejected. The next was **ETRA** and again, this was rejected as it was not in the original **Flight Plan**. The next was **DVL** and this *was* in the original **Flight Plan** but not the **DVL** the system was expecting; the system spotted that the **DVL Waypoint** was *before* the UK entry point and so couldn't possibly be a viable UK exit point so the system raised a critical exception requiring manual intervention. The backup system then took over and repeated the processing resulting in the same conclusion – a critical exception, thus bringing all processing to a standstill. Note that this is a subtle problem; it wasn't that there were two **DVL Shortname**s in the original **Flight Plan** but rather, the system processing speculating on whether the French **DVL** was the intended exit point was tripped up by the earlier occurrence of a **Waypoint** with the same name.

From this, I'm sure you'll agree that allowing duplicate **Waypoint**s exposes systems to downstream problems, even when it is known that they exist. Is **DVL** an isolated edge case of duplicate **Waypoint**s? Sadly not; there are almost 4,000 duplicate **Waypoint**s globally and some like **BI** have more than 10 duplicated around the world!

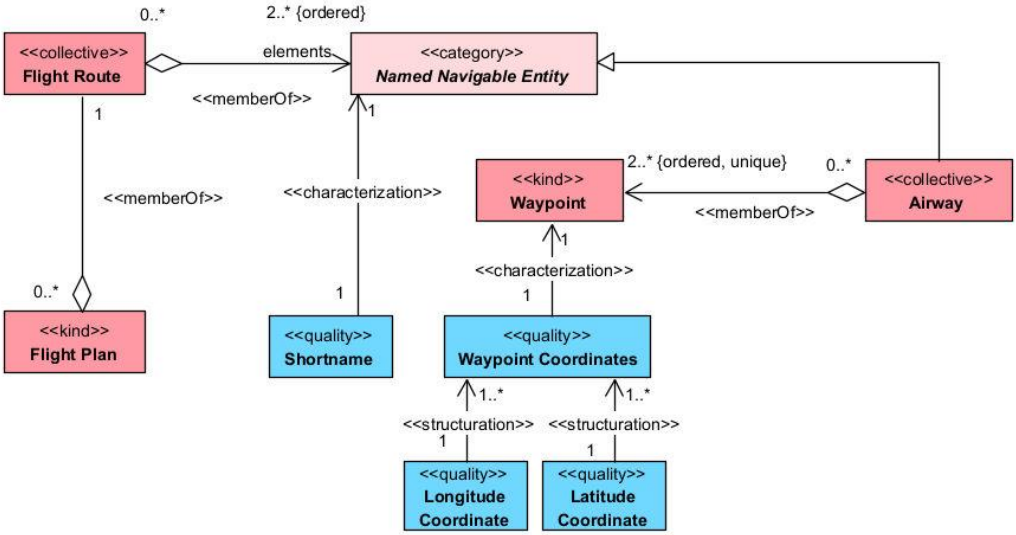
There are many reasons why duplicate **Waypoint**s are tolerated now, which we don't have time to explore, but we'll close by simply asking how would we change our model to enforce uniqueness of **Shortname**s (across **Waypoint**s and **Airway**s)?

Similar problems are encountered in modelling and modern web-based systems (addressed through namespaces and Uniform Resource Identifiers). A variety of patterns have been developed and road tested to allow the same label to be used to identify multiple things without losing uniqueness.

In our case, we could try to prevent duplicates by using multiplicities, but setting the **Waypoint** to **Shortname** and **Airway** to **Shortname** relationships to 1 on both sides would not prevent a **Waypoint** and an **Airway** sharing the same **Shortname.** A neater solution is to add an abstraction of **Waypoint** and **Airway** and then add the multiplicity constraint to that higher **abstract Class**.

To do that, we will generalise the two **Class**es into an **abstract Class Named Navigable Entity**. An **abstract Class** in UML is used for things that cannot be directly instantiated, or in other words only concrete subclasses may be instantiated. Familiar examples include **Mammal**, **Fuel**, **Molecule**. In our model, as the two subtypes are different types (**Collective** and **Kind**), we introduce our final new UFO term: **Category**. **Category** can be used a little like a **Mixin** that generalise **Class**es with no common identity principle but where the underlying **Class**es are all **Rigid**; that is, **Airway** and the **Waypoint** are both a **Named Navigable Entity** for their entire lifetime. (Remember, the **Mixin Insurable Item**s generalised some **Anti-Rigid Class**es such as **Pilot** – a **Phase** a **Human** may go through for part of their life).

The change is shown below again, with some diagram elements omitted for clarity.



A fairly straightforward model change that alleviates a lot of potential problems downstream!

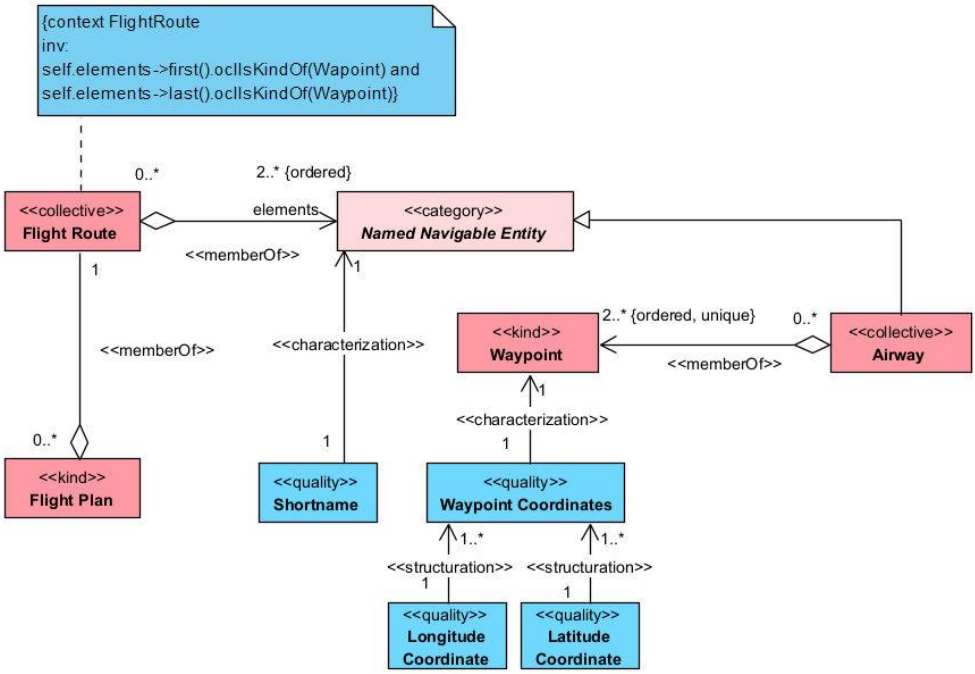
Note that we do now have a small wrinkle that we should address. In the model above, we could have a **Flight Route** formed only of **Airway**s. However, a **Flight Route** must[2] actually start and end at an airport, aerodrome or other location designated by a **Waypoint**.

Since **Airway**s never include take-off and landing point, a **Flight Route** must always start and end at a specific **Waypoint** and might (but not always) contain one or more **Airway**s between those points. To resolve this, we are going to use a little OCL again on a different **Class**.

In the following diagram you will see that a new OCL constraint applies to the **Flight Route**:

```
context FlightRoute
inv:
self.elements->first().oclIsKindOf(Waypoint) and
self.elements->last().oclIsKindOf(Waypoint)
```

This is saying that in the `context` of a **Flight Route**, the first and last element must always be a single **Waypoint**. Note that, importantly, due to the way the UML language works, this constraint does not mandate that start and end **Waypoint**s be different[3] – if we did this, then it would prevent us modelling flights that take-off and land at the same place.



[2] This is a slight oversimplification which hopefully can be accepted for the purposes of this article. For example, military or emergency vehicles can potentially take off from unmapped fields.

[3] UML does allow **Association End**s to be marked `{unique}` in cases where this is desirable.

Summary

In this episode we have introduced a few more concepts such as the idea of constraints that are applied using an Object Constraint Language (OCL) and the new stereotype called **Category**, which is **Mixin**-like but used only for **Rigid** **Class**es with shared properties. We also saw how to expand our model when new concepts become apparent, like **Airway**. These have also helped us resolve the couple of outstanding questions we had on the model that we developed from the three previous episodes.

By exploring the NATS outage example, we've seen that despite duplicate **Waypoint**s being well-known and tolerated, the devil is indeed in the detail and it is difficult to ensure that all downstream processing caters for every possible eventuality. This really underlines the importance of using ontologies (ideally at the outset): by specifying concepts in this formal way, it removes the possibility of ambiguity and the risk of misinterpretation. Initially, the model may not perfectly reflect the real-world intent, but as a means of reasoning about concepts and relationships it provides the platform for identifying shortcomings and to evolve the model so it becomes a faithful representation. With a formal ontology, it should also be easier to assess the impact of change when new concepts are introduced or existing concepts need refinement.

We really hope you have found this series of articles interesting and, of course, hope you've learned something about ontological modelling! If you would like more information on ontologies or indeed the work of the OWG then please get in touch (details are on the SCSC website: scsc.uk/owg)

References

[1] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, in Knowledge Acquisition, 5(2):199-220, Academic Press, 1993, http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[2] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, T. P. Sales, Ufo: Unified foundational ontology, Applied ontology 17 (2022) 1–44, https://doi.org/10.3233/AO-210256

[3] OntoUML Specification, Marek Suchánek, https://ontouml.readthedocs.io/, accessed Jan 2026

[4] Unified Modelling Language, https://www.omg.org/spec/UML

[5] OWG article "Modelling Tools", https://scsc.uk/working-groups/ontology-wg-members-area/tooling

[6] RelOver anti-pattern, OntoUML Anti-Pattern Catalogue, https://ontouml.readthedocs.io/en/latest/anti-patterns/RelOver/index.html, accessed July 2026.

[7] Air traffic shutdown sparks NATS homeworking change, https://www.bbc.co.uk/news/articles/cev9l3dekkko accessed July 2026.

[8] CAA, NATS Major Incident Investigation Final Report, Flight Plan Reception Suite Automated (FPRSA-R), Sub-system Incident 28th August 2023, CAP2993C 14 Nov 2024.

[9] Benevides, A.B., Guizzardi, G., Braga, B.F.B. and Almeida, J.P.A. (2010) 'Anti-Patterns in Ontology-Driven Conceptual Modeling', in Conceptual Modeling – ER 2010, Lecture Notes in Computer Science (LNCS), vol. 6415. Berlin, Heidelberg: Springer, pp. 317–331. https://doi.org/10.1007/978-3-642-16372-2_23

Image attributions:
leading image: Richard N Horne, CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0, via Wikimedia Commons
original planned route: Map data provided by SkyVector (www.skyvector.com) / FAA Aeronautical Information Services.
UK waypoints: NATS Major Incident Investigation Final Report (CAP2993C), UK Civil Aviation Authority, November 2024. Reproduced under Open Government Licence

The Ontology Working Group (OWG)



The OWG was set up in 2020 to help address issues with the lack of formalisation in terms and conceptual relationships used in SCSC standards and guidance documents. In particular, the similar, but at times different, interpretations of terms and concepts used between safety and security domains is an ongoing cause of confusion and misunderstandings and presents barriers to more formal integration of these disciplines.

The OWG has its origins in the Data Safety Initiative Working Group (DSIWG), which, between 2016 and 2019, explored the development of an ontology model for the concepts described in the Data Safety Guidance (DSG). However, it was realised that the work couldn't be completed without a wider ontological model for risk, and so the OWG was established and is currently working on developing a clear, systematic terminology, and unambiguous model of risk and value concepts and their relationships.



Seminar: How to Make the Most of AI: A Workshop for Safety Engineers



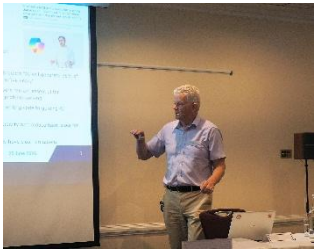
The "How to Make the Most of AI: A Workshop for Safety Engineers" seminar was held at the Hilton Euston hotel in London, on 25th June 2026[4] and was hosted by Mike Parsons. The speakers discussed a range of techniques and insights into how safety engineers might benefit from the use of generative Artificial Intelligence (AI) / Large Language Models (LLMs), examining everything from prompt engineering to building applications and custom tools from AI code generators.

The four speakers presented interesting and lively sessions, all of which generated great discussion. (Note that AI is used broadly to include all aspects such as LLMs and Machine Learning). Topics included:

- Alan Simpson (Ebeni): **Beyond the Hype: LLMs in Safety-Critical Engineering - Principles, pitfalls and the changing role of the engineer** – what AI will mean for safety engineers, how we can ensure that we continue to use AI safely and appropriately.
- Rudi (Redford-Brown) Hennessy (NHS England) **Prompting LLM to support hazard assessment** – the use of AI in safety-critical engineering within the healthcare sector, expectations and consequences
- Paul Hampton (Consultant and SCSC): **The Unreliable Guru** – using AI to rapidly generate code exploring some real-world examples
- Ben Fulford – (BMT): **Putting AI to Work: From Better Prompts to Building Your Own Tools** – practical techniques for using AI, how to get started, prompt engineering and tool building

[4] This was an incredibly hot day and we were very grateful to all those who attended in person!

Alan Simpson – Beyond the Hype: LLMs in Safety-Critical Engineering - Principles, pitfalls and the changing role of the engineer



Alan began by defusing some of the more extreme mainstream fears: namely, reassuring the audience that AI is not going to take over the jobs of safety engineers! The audience contributed by providing use cases for how safety engineers are already using AI, and Alan raised some important points about these, including the potential for untrustworthy outputs. He pointed out that the safety engineering challenge is how to justify with sufficient confidence, that the combination of AI tool, human oversight and process controls is appropriate for the consequences of being wrong.

AI tools, Alan reminded us, can fail in very different ways to traditional tools, and the real challenge is in justifying the use of the AI tools with sufficient confidence for the environment of use. He warned us of some of the risks of generative AI, including hallucination risk, context misunderstanding, risk of over-reliance / de-skilling and the potential for malicious compromise. He also explored the trust paradox: the greatest risk from AI may be the gradual erosion of scrutiny which we apply to its answers. Alan wrapped up with a discussion of how we might reframe the question using standards and the outputs from the SCSC AI Working Group (https://scsc.uk/saias), building on ISO 42001. The takeaway message was that AI – when it comes to safety engineering – is not wholly good, bad (or even ugly!) It is a tool that offers significant benefit but must be used only when we have sufficient confidence and sound engineering judgement in its capability.

Rudi (Redford-Brown) Hennessy (NHS England) Prompting LLM to support hazard assessment – Practical Gains, Real Risks



Next up was Rudi, who took us through an introduction of what AI could usefully be used for within healthcare, including drafting, summarising, research, process automation and ideation, or helping us brainstorm. Within safety specifically, she identified AI to be genuinely useful in hazard log drafting, standards research and incident analysis – all very complex fields, where many safety engineers would appreciate AI serving as a "first pass"

Rudi next took us through a practical workflow for AI assisted hazard identification, which prompted much audience interest and a number of questions. Many of these were answered by her next topic, a discussion of the AI risks to consider and precisely what these might mean in terms of healthcare. This was reinforced by her emphasis that AI is a supplementary tool and its primary value lies in performing the more routine tasks and "giving us time to think". AI, she emphasised, is not a replacement for clinical safety expertise, but it can help healthcare safety engineers to focus their efforts on the areas that most need human judgement and assessment.

Paul Hampton (Consultant and SCSC): The Unreliable Guru

Paul considered the use of AI specifically for generating code and software engineering and asked us: can AI be the software "guru", consolidating creativity, art, and effective use of language to produce code? He let us into the secret that much of the technical entertainment material for SSS'25 and SSS'26 was produced with human-AI complementary effort. The example quiz question requiring the audience to identify the safety concept referred to in an AI-remixed version of ABBA's Waterloo proved very popular!



Paul also identified AI was very helpful in checking code that he had written, noting that when he asked it questions about whether code might fail in one particular way (failing to exit) not only did it confirm this, but identified other concerns. This was, he considered, an example of Guru-level coding in terms of the suggestions that it provided. Nevertheless, the "nearly there" rabbit hole – when you get almost what you want, but not quite! – remains a significant concern. AI, he reminded us, can often produce answers that are almost correct, but crucially, not quite. A better classification of it might be that it is an "unreliable guru", in that it is functional for well-defined problems but not for open problems. He then demonstrated the power of AI Integrated Development Environments by showing how Anysphere's Cursor application generated a fully functioning Boeing 737-800 Take-off Speed Calculator within 25 minutes. He finished with some salutary reminders that AI might be better, faster and more accurate than human developers but – again echoing the theme of the day – could not be used as a replacement in safety-critical systems engineering (yet!) Ultimately, accountability must remain with human engineers, who must understand, justify and take ownership of every AI-generated component before it is incorporated into a deployed system.

Ben Fulford - Putting AI to Work: From Better Prompts to Building Your Own Tools – Getting More from AI

The last of our talks was from Ben Fulford, presenting a practical route to AI integration for safety engineers including prompt engineering and tool building. He set out the four rungs of the ladder: prompting well, context engineering, orchestration, pay-off and then recapped the BMT prompt framework for effective LLM prompting. He emphasised the importance of pressure-testing the output, and also of providing context: give the AI your standards, your safety case and your hazard log. This provides the necessary *context engineering*, the second rung on the ladder.



Moving onto the third rung – orchestration – Ben invited us to consider this as creating a board of advisors who disagree. This involves running several models, each given a different job, and intended to challenge each other.

This should involve human-in-the-loop contribution, but this does not necessarily have to mean a human reviewing the AI code! Instead, Ben suggested that we consider the use (by a human) of a second model to perform the verification, checking generated code against the specification. Finally, rung 4 – the payoff – where Ben illustrated this by reference to a narrative game he's been involved in creating, where the safety engineer is the hero! Certainly, an attractive thought, at least judging by the audience reaction. Ben wrapped up with some concrete and practical suggestions for steps we might take to start using the prompting framework and techniques he had introduced.

Workshops

The afternoon saw interactive workshops, with Alan, Rudi, Paul and Ben all leading sessions on how to incorporate and begin using AI. The first set of workshops involved the topics described below:

Example hazard analysis of a fictitious Autonomous Rail Inspection Drone



This workshop kicked off with an interesting use of ChatGPT: one participant had copied the seminar description into ChatGPT to ask what might be a valuable output from the task. Immediately the participants realised it had folded much of the written information about the workshops together – an example of why human insight is vital!

Working together with the AI, the participants were able to unpick the task and begin creating the hazard analysis by using their own experience of similar artefacts. Questions raised were how to define hazards to the AI, and how to establish the level of hazard and the depth of the analysis required. The AI-human collaboration performed well, with the team identifying 45 gaps in the system description that it was given.

2 out of 3 Voting Logic Module with Dynamic Sensor Health Tracking and Rate-of-Change Limiting



This workshop dived right into addressing the problem, with the facilitator demonstrating how quickly AI can generate a fully functioning 2 out of 3 Voting Logic Module from a one-page requirements specification. Participants explored hazards in depth, including those related to the rate of change – and the requirement for the code to flag significant changes in a sensor's value as indicative of an error – as well as considering erroneous rates of change, and the requirement for the AI to monitor this. One issue raised was that due to the need to manage the context window size, every new AI agent was required to begin from first principles: it had no knowledge of previous agent chats, and therefore couldn't use previous context (which has both advantages and disadvantages!) The AI was reasonably effective at assessing the adequacy of its code, but appeared to require significant prompting to refine the solution.

GUARDIAN: a preliminary hazard list for a safety programme



This group began similarly to the others, by feeding the workshop topic into AI and asking it to produce hazard examples for a draft system. Participants also discussed the potential for AI use in this example to have associated hazards including data leakage and privacy compromise.

An excellent example of how human expertise is essential for collaboration! One point that was particularly noted was the benefit of having (human) coding expertise within the room. Participants also began to identify a preliminary hazard list with the help of AI, and multiple participants identified that the use of two models was beneficial so that the same task could be run through both to provide assurance. The team also expressly asked the AI to act as a red team to challenge and interrogate the output from the team.

Hazard examples for Ambient voice technology / AI discharge summaries



This group focused on the healthcare domain, and immediately noted from their first prompt – asking AI for hazard examples – that the system clearly lacked system context.

Human context was also identified as significant, including the relevance of patient behaviour. Participants then assessed the adequacy of an AI-produced hazard log (after identifying that this had originated in an NHS England hazard log!)

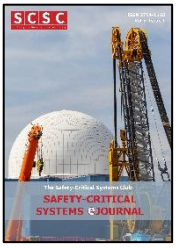
Participants also found that when they provided context, including relevant standards, it produced a more comprehensive and defensible hazard log without the need for amending or refining the prompt. Participants also experimented with providing the AI with the specific system under investigation, and identified that prompt engineering was of significant benefit with more context provided.

Panel discussion and wrap up

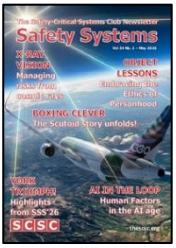
The workshops concluded with a panel discussion with the speakers, and audience questions. The topics were wide ranging, including more case studies of how people were using AI or potential future use cases (the dreaded "thousand-word email" example came up here!) The conclusion overall was that AI and humans make great collaborative partners but that for safety critical work the AI should always be that: a partner and not a leader.

Report by Catherine Menon

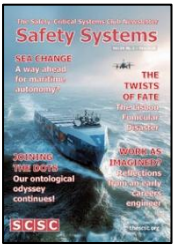
Recent Safety Publications



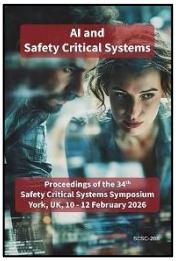
Safety-Critical Systems eJournal vol.5 no.1 2026 First Edition
scsc.uk/scsc-214



The Safety-Critical Systems Club Newsletter, Volume 34, Number 2 scsc.uk/scsc-212



The Safety-Critical Systems Club Newsletter, Volume 34, Number 1
scsc.uk/scsc-209



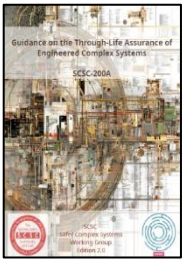
Proceedings of the 34th Safety-Critical Systems Symposium
scsc.uk/scsc-208



Safety-Critical Systems eJournal vol.4 no.2 Winter Issue 2025
scsc.uk/scsc-210



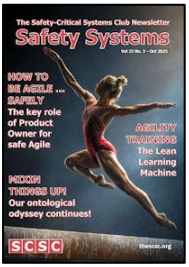
Safety-Critical Systems Data Safety Guidance v4.0 Parts 1, 2 and 3
scsc.uk/scsc-127K



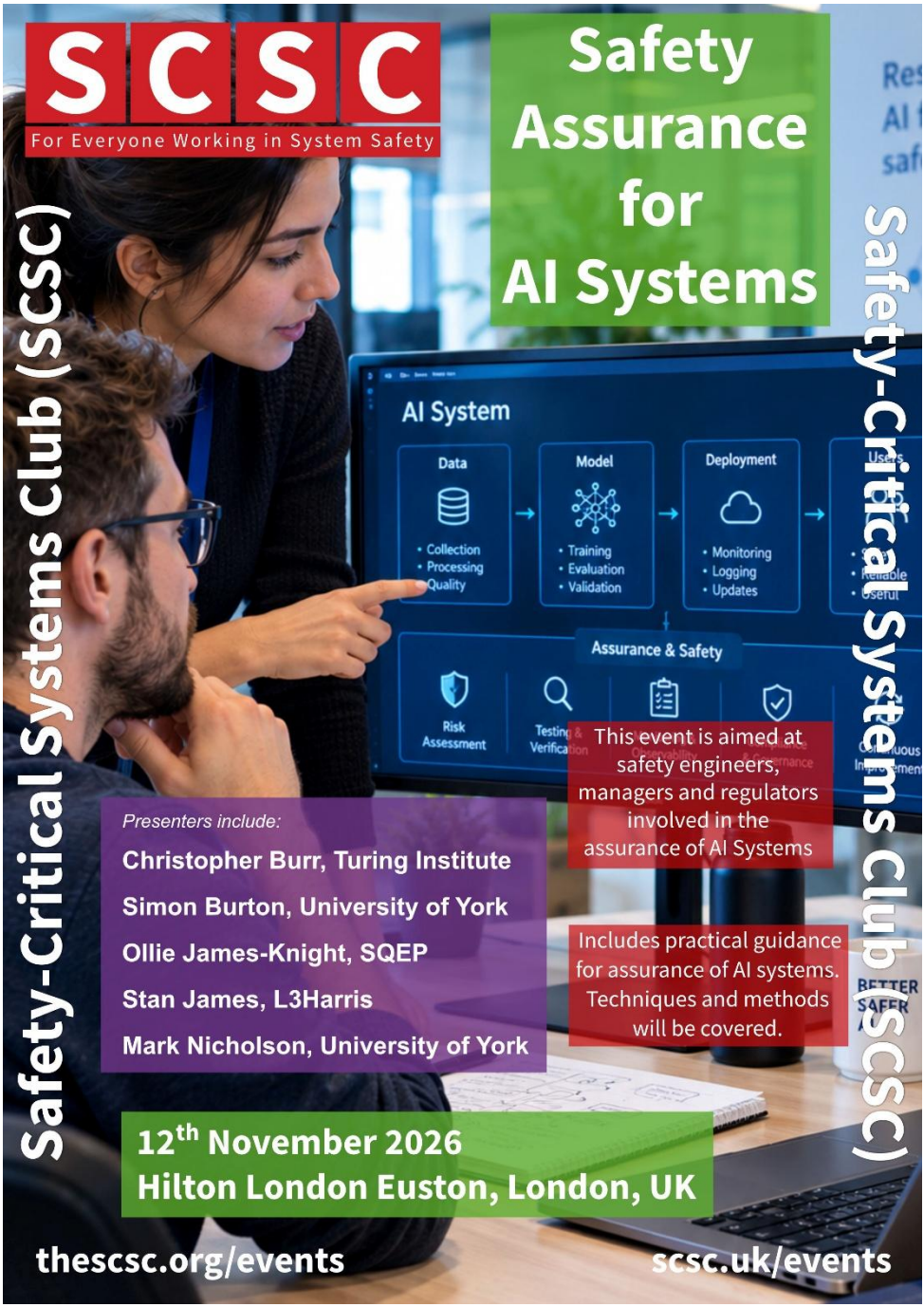
Guidance on the Through-Life Assurance of Engineered Complex Systems v2.0
scsc.uk/scsc-200A



Service Assurance Guidance v4.1
scsc.uk/scsc-156E



The Safety-Critical Systems Club Newsletter, Volume 33, Number 3
scsc.uk/scsc-206


SCSC
For Everyone Working in System Safety
Safety Assurance for AI Systems
Safety-Critical Systems Club (SCSC)
Safety-Critical Systems Club (SCSC)
AI System
Data
• Collection
• Processing
• Quality
Model
• Training
• Evaluation
• Validation
Deployment
• Monitoring
• Logging
• Updates
Assurance & Safety
Risk Assessment
This event is aimed at safety engineers, managers and regulators involved in the assurance of AI Systems
Presenters include:
Christopher Burr, Turing Institute
Simon Burton, University of York
Ollie James-Knight, SQEP
Stan James, L3Harris
Mark Nicholson, University of York
Includes practical guidance for assurance of AI systems. Techniques and methods will be covered.
12th November 2026
Hilton London Euston, London, UK
thescsc.org/events
scsc.uk/events

60 Seconds with … Tim Rowe



After an electronics apprenticeship with UK NATS and a couple of years fixing radar displays, Tim got release to university to study Electrical and Electronic Engineering. From there he went on to line management of airport ground systems and then to the NATS safety management team.

He subsequently moved to a succession of consultancy companies before setting up as an independent consulting engineer. He obtained long service awards from two of his employers, demonstrating his ongoing commitment to avoiding job interviews, and picked up degrees in Computing for Commerce and Industry and Humanities with English Language along the way.

He thought he retired a few years ago, but a former client persuaded him to come out of retirement for another job, so it turns out he is still available for work if it looks sufficiently interesting and doesn't involve too many early mornings. He is active in SCSC working groups, particularly the Data Safety Initiative Working Group and he was instrumental in restructuring and editing version 4 of the Data Safety Guifance (all three parts!)

What first attracted you to working in the field of System Safety?

It was by accident! I had a natural flair for breaking things from an early age. When acceptance testing of new equipment included free testing, I turned out to be good at exposing weaknesses. My team thought that put me in a good position to write the safety cases for the equipment. Of course, there's not much connection between those two things, but there I was working out how to manage system safety and discovering that I enjoyed it. And I like to think I turned out to have some aptitude.

What aspect of your career are you most proud of?

That I have no fear of flying. Obviously, that's mainly not me; aviation safety is a huge and complex affair. But I'm proud to be even a small part of it.

Tell us one interesting unusual fact about yourself!

Everything about me seems ordinary to me! People used to be surprised to see me doing embroidery in the crowd at music festivals, but nobody seems to find that unusual now.



What advice would you give to yourself age 12?

It gets better.

What future changes would you like to see in the field of System Safety?

I'd like to see a widening of the idea of "safety" from the narrow focus it tends to get of just physical harm to the person (and in some fields, the environment). But hazard identification can identify financial, political, reputational effects and more that it would benefit an organisation to address. Whether that's by taking safety to include a wider range of harms or just improving communication between the safety team and other teams, I think there's a lot that could be gained.

What's your most favourite quote or motto?

I have a T-shirt with the text "*If only reading burned more calories…*"

If you could learn to do anything, what would it be?

It would be great to learn to play a few more of the musical instruments I keep buying at music festivals.

If you could be any fictional character, whom would you choose?

Carrot Ironfoundersson, from Terry Pratchett's "Discworld" series. Everyone knows he's the rightful king, but he doesn't try to claim the title because he knows he can do more good as a captain in the city watch. I like that attitude. I'm also jealous of the easy and natural way he gets to know everyone and gets on with people.

"One of the biggest challenges with safety engineering has always been social, or political if you like: persuading the right people that safety management is actually needed at all"

How do you see the rise of Artificial Intelligence affecting the future of safety engineering?

My first reaction is to ask, "which of the many, completely different, technologies marketed as AI? But on reflection, it doesn't matter much. All forms of AI can only act as part of a system – even agentic AI needs a connection to the outside world. And all of them can give wrong results (some AI technologies more often than others). And in safety engineering, we're used to building acceptably safe systems out of elements that are to some extent and in some ways unreliable. AI technologies might change the ways and the extent, but I think we'll still be doing what we've always been doing. And one of the biggest challenges with safety engineering has always been social, or political if you like: persuading the right people that safety management is actually needed at all. I don't see any signs of that challenge going away either. Plus ça change, plus c'est la même chose.

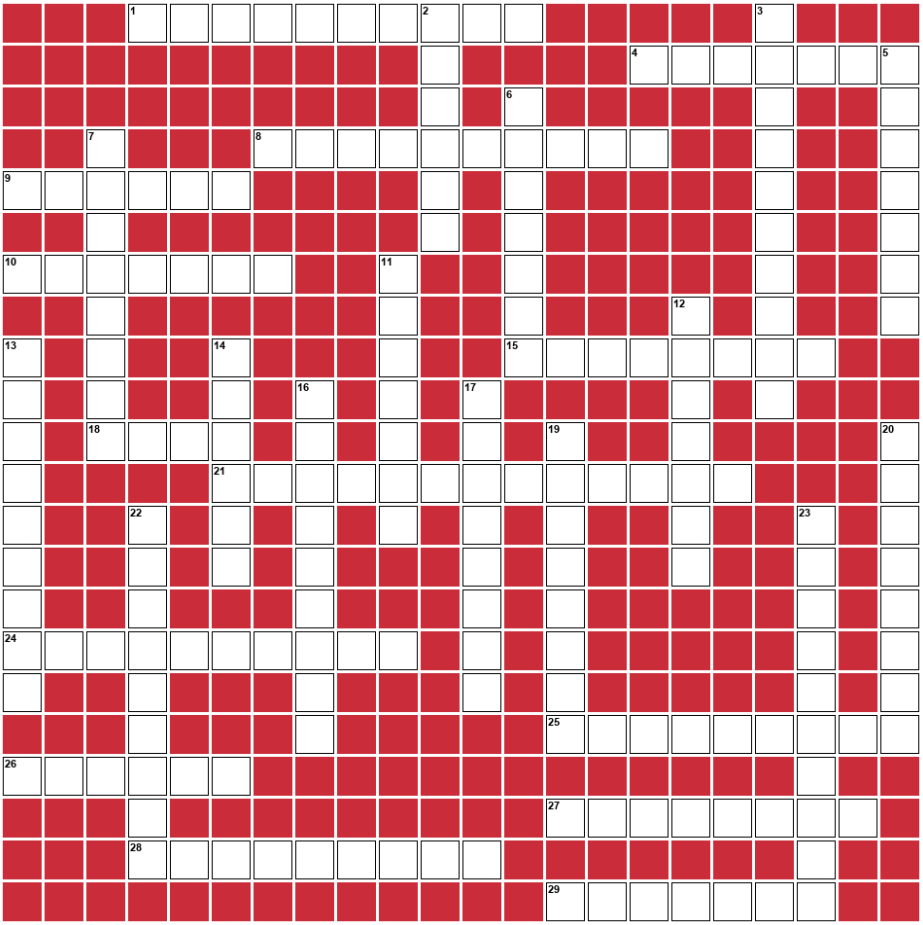
Tim can be contacted at digitig@gmail.com if you want to discuss anything from work to embroidery patterns!

Safety Systems Crossword

Welcome to the 5th Safety Systems Crossword prize competition where you can test your knowledge of the world of System Safety!

Send you completed grids to crossword@scsc.uk or drop into the crossword entry box at SSS'27 by 1pm on 10th Feb 2027.

The first correct answer drawn from the box at SSS'27 will win a £100 gift voucher!



Across

1. City of a 2025 cargo-aircraft crash shortly after takeoff. (10)
4. European electrotechnical body behind railway standards EN 50126, 50128 and 50129. (7)
8. Type of failure arising from design or specification error rather than random hardware. (10)
9. Name of the Lisbon funicular whose connecting cable failed in 2025. (6)
10. Somerset site of the UK's first new nuclear power station in a generation. (7)
15. Virtual boundary that constrains where an autonomous vehicle may operate. (8)
18. Abbreviation for an accident in which an airworthy aircraft is flown into terrain. (4)
21. Generation of plausible but false output by a large language model. (13)
24. Contact of an aircraft's rear fuselage with the runway during takeoff or landing. (10)
25. Anthropic programme that restricted a model to defensive cybersecurity partners. (9)
26. Diagram showing threats, a top event, consequences and barriers. (6)
27. Risks that are intrinsic to an endeavour itself. (8)
28. Cruise ship that capsized after striking rocks off Italy. (9)
29. Impairment of performance associated with insufficient rest. (7)

Down

2. Type of condition that remains dormant until combined with other factors. (6)
3. Provision of more than one means of performing a required function. (10)
5. Type of failure in which one loss precipitates further losses. (7)
6. Deliberate radio interference that denies use of a navigation signal. (7)
7. Type of inter-species virus transmission. (8)
11. Familiar shape of a reliability curve over a product's life. (7)
12. Location of a 2026 fatal collision between two passenger trains. (7)
13. Use of different means to achieve the same function to reduce common-cause failure. (9)
14. Large language model whose public release was delayed over misuse concerns. (6)
16. Marine animal known to cause shutdowns of nuclear power plants. (9)
17. Design property that a system moves to a safe state on loss of power or control. (8)
19. Timer used to detect that a computer has stopped responding. (8)
20. Transmission of false navigation signals to deceive a receiver about its position. (8)
22. Form of argumentation balancing opposing positions. (9)
23. Ability of a system to adjust functioning before, during or after a disturbance. (10)

Name: _____

Contact Nos/email: _____

Connect

The Newsletter and eJournal

Do you have a topic you'd like to share with the systems safety community? Perhaps an interesting area of research or project work you've been involved in, some new developments you'd like to share, or perhaps you would simply like to express your views and opinions of current issues and events. There are now two publishing vehicles for content – shorter, more informal content, can be published in the Newsletter with longer, more technical peer-reviewed material more suitable for the eJournal. If you are interested in submitting content, then get in touch with Paul Hampton for Newsletter articles: paul.hampton@scsc.uk or John Spriggs for eJournal papers: john.spriggs@scsc.uk

The SCSC Website

Visit the Club's website thescsc.org for more details of the Safety-Critical Systems Club including past newsletters, details of how to get involved in working groups and joining information for the various forthcoming events.

Facebook

Follow the Safety-Critical Systems Club on its very own Facebook page.

www.facebook.com/SafetyClubUK

X/Twitter

Follow the Safety-Critical Systems Club's X/Twitter feed for brief updates on the club and events: @SafetyClubUK

LinkedIn

You can find the club on LinkedIn. Search for the Safety-Critical Systems Club or use the following link:

www.linkedin.com/groups/3752227

Advertising

Do you have a product, service or event you would like to advertise in the Newsletter? The SCSC Newsletter can reach out to 1,000's of individuals involved in Systems Safety and so is the perfect medium for engaging with the community. For prices and further details, please get in touch with the Newsletter Editor.

SCSC Working Groups


Security Informed Safety


Assurance Cases


Service Assurance


AI / Autonomous Systems


Data Safety Initiative


Safer Complex Systems


Multicore


Safe System Architecting


Safety Culture


Ontology



Safety Futures Initiative


Systems Approach to Safety of the Environment


Safety Management & Safety Management Systems

The Safety-Critical Systems Club is committed to supporting the activities of working groups for areas of special interest to club members. The purpose of these groups is to share industry best practice, establish suitable work and research programmes, develop industry guidance documents and influence the development of standards.

Security Informed Safety



The Security Informed Safety Working Group (SISWG) aims to capture cross-domain best practice to help engineers find the 'wood through the trees' with all the different security standards, their implication and integration with safety design principles to aid the design and protection of secure safety-critical systems and systems with a safety implication.

The working group aims to produce clear and current guidance on methods to design and protect safety-related and safety-critical systems in a way that reflects prevailing and emerging best practice.

The guidance will allow safety, security and other stakeholders to navigate the different security standards, understand their applicability and their integration with safety principles, and ultimately aid the design and protection of secure safety-related and safety-critical systems.

Last year, the working group published its first piece of guidance: Co-Assurance of Safety and Security.

Lead **Stephen Bull** stephen.bull@scsc.uk

Assurance Cases

The Assurance Cases Working Group (ACWG) has been established to provide guidance on all aspects of assurance cases including construction, review and maintenance. The ACWG will:

- Be broader than safety, and will address interaction and conflict between related topics
- Address aspects such as proportionality, rationale behind the guidance, focus on risk, confidence and conformance
- Consider the role of the counter-argument and evidence and the treatment of potential bias in arguments



One of the working group's activities is the maintenance of the Goal Structuring Notation (GSN) Community standard. See scsc.uk/gsn for further details. This includes a library scsc.uk/library of relevant papers previously published by Tim Kelly, which are now hosted on the SCSC website.

In May 2021, the group published v3.0 of the standard: scsc.uk/scsc-141C

In Aug 2021, the group published v1.0 of the Assurance Case Guidance: scsc.uk/scsc-159

Lead **Jane Fenn** jane.fenn@baesystems.com with support from **Phil Williams** phil.williams@scsc.uk

Service Assurance



Risks presented by safety-related services are rarely explicitly recognised or addressed in current safety management practices, guidelines and standards.

It is likely that service (as distinct from system) failures have led to safety incidents and accidents, but this has not always been recognised. The Service Assurance Working Group (SAWG) has been set up to produce clear and practical guidance on how services should be managed in a safety-related context, to reflect emerging best practice.

The group is working on an update to the guidance v4.1 hopefully to be published in 2026.

Lead **Kevin King** kevin.king@baesystems.com

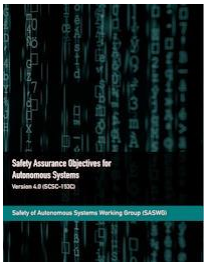


Safety of AI / Autonomous Systems



It is clear that AI and autonomous systems can introduce many new paths to accidents, and that autonomous system technologies may not be practical to analyse adequately using accepted current practice. Whilst there are differences in detail, and standards, between domains many of the underlying challenges appear similar and it is likely that common approaches to core problems will prove possible.

The SCSC Safe AI Working Group (SAIWG) aims to capture cross-domain best practice and guidance on key topics within the design, evaluation, assurance, and approval of safety systems that use or are developed using AI, bringing together emerging standards and key results from the incredible amount of research being conducted into AI safety.

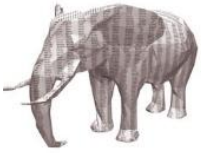


The working group was kicked-off at SSS'24 and is led by Alan Simpson. The SAIWG will conduct regular meetings, workshops, and publications to share knowledge and experience on various topics related to AI and safety systems, such as coordination of safety with other disciplines, evaluation of risk, and mapping of terminology and language.

The group builds on the earlier work of the Safety of Autonomous Systems Working Group (SASWG) that culminated in production of the 4th version of its guidance: Safety Assurance Objectives for Autonomous Systems, in Feb 2024 scsc.uk/scsc-153C

Lead Alan Simpson alan.simpson@ebeni.com

Data Safety Initiative



Data in safety-related systems is not sufficiently addressed in current safety management practices and standards.

It is acknowledged that data has been a contributing factor in several incidents and accidents to date and there is foreseeable harm that can arise from Machine Learning and Large Language Models' use of data by Artificial Intelligence (AI) systems that are subject to issues of biasing, interpretation and, arguably, falsification. There are clear business and societal benefits, in terms of reduced harm, reduced commercial liabilities and improved business efficiencies, in investigating and addressing outstanding challenges related to safety of data.

The Data Safety Initiative Working Group (DSIWG) aims to have clear guidance on how data (as distinct from the software and hardware) should be managed in a safety-related context, which will reflect emerging best practice.



An update to the guidance (v4.0) was published in Jan 2026: scsc.uk/scsc-127K This was a major release for the guidance and saw the original document reorganised into three separate parts that can be upissued independently. The group is thriving having celebrated its 100th meeting in September!

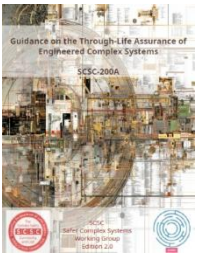
Lead Mike Parsons mike.parsons@scsc.uk

Safer Complex Systems



The Safer Complex Systems Working Group (SCSWG) builds on the IET/RAE work already done in this area. It is recognised that the RAE work is ongoing and collaboration is encouraged. The group's mission is to produce practical guidance on developing, managing and assuring complex systems throughout their lifecycle (so as to achieve and justify their safety). The following provides a problem statement:

- It is acknowledged that complex systems are becoming more prevalent with more opportunities to cause harm
- There are also new complex systems arising from using combinations of existing systems and services which are then used for safety purposes
- Complex systems are not sufficiently addressed in current safety management practices and standards
- In particular, complex interactions and emergent behaviours are not currently assessed and managed sufficiently
- There could be benefit in developing new analysis, tools and techniques to manage complex system risks
- There are clear business and societal benefits, in terms of reduced harm, reduced liabilities and improved business efficiencies, in improved management of complex systems risk

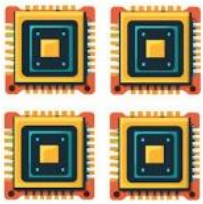


The group produced the second edition of Guidance on the Through-Life Assurance of Engineered Complex Systems (scsc.uk/scsc-200A) in January 2026. Note that the title now includes the word "Engineered" to be specific about the types of system in scope.

Contact the group lead if you would like to attend future meetings or find out more.

Acting Lead Mike Parsons mike.parsons@scsc.uk

Multi- and Manycore Safety



It is becoming harder and harder to source single-core devices and there is a growing need for increased processing capability with a smaller physical footprint in all applications. Devices with multiple cores can perform many processes at once, meaning it is difficult to establish (with sufficient evidence) whether or not these processes can be relied upon for safety-related purposes.

Parallel processes need to access the same shared resources, including memory, cache and external interfaces, so they may contend for the same resources. Resource contention is a source of interference which can prevent or disrupt completion of the processes, meaning it is difficult to know with a defined uncertainty the maximum time each process will take to complete (Worst Case Execution Time, WCET) or whether the data stored in shared memory has been altered by other processes. The Multi- and Manycore Safety Working Group (MCWG) has been established to explore the future ways of assuring the safety of multi- and manycore implementations.

Lead Lee Jacques Lee.Jacques@leonardocompany.com

Safe System Architecting



The SCSC Safe System Architecting Working Group (SSAWG) is a joint initiative between the INCOSE UK Architecture Working Group and the Safety Critical Systems Club, with a vision of addressing systems safety through the promotion of choosing, using, and developing appropriate architecture. Two focal areas are defined as:

- Safety-driven architecting – what safety drivers are likely to exist when architecting is being considered
- Architecting system safety – what architectural factors may realise, enable, support, or preclude the achieving of safety considerations

The groups Mission Statement is:

- To produce practical guidance on addressing system safety during system architecting.
- To encourage and facilitate collaboration between INCOSE and the club.

The working group meets every 6 weeks online via Teams. Note that the INCOSE UK AWG meets every quarter.

If you would like to be involved in the group, please contact the co-chairs.

Co-chairs: Siyuan Ji s.ji@lboro.ac.uk and **Jane Fenn** jane.fenn@baesystems.com

SCSC Safety Culture



The Safety Culture Working Group (SCWG) has been established to provide guidance on creating and maintaining an effective safety culture. The group seeks to improve safety culture in safety-critical organisations focussed on product and functional safety, by sharing examples and latest approaches collated from real-life case studies.

Meetings provide an opportunity to discuss any particular aspects attendees are interested in taking forward, and to help set future directions for the group.

In Dec 2023, the group published a position paper for assessing and managing safety culture. scsc.uk/r2995

A seminar was held 19th June 2025 in London and focussed on how Safety Culture practices need to change to accommodate Artificial Intelligence. SCSC members can access the seminar presentations and videos here: scsc.uk/e1156

Lead Anne Seldon anne.seldon@wae.com

Ontology



The Ontology Working Group (OWG) develops ontologies that will form the basis of SCSC guidance, as well as having wider industrial and academic applications.

During system development, and especially with Model-Based Systems Engineering (MBSE), it is essential to build a model of the world that the system will inhabit, and this involves developing and structuring concepts, terms and their relationships. An ontology is an explicit specification of such a conceptualization and provides the foundation for MBSE and architectural descriptions in general. Ontologies are also increasingly important, if not essential, for explainable Artificial Intelligence (AI), enterprise knowledge systems and the standards that underpin them.

The OWG is currently working on the definition of an ontology of risk and value for application in guidance for risk-based decision making – notably safety and security. The framework for modelling is the Unified Foundation Ontology (UFO) and OntoUML, a domain-specific language (DSL) for modelling in UML tailored for ontological modelling based on UFO.

The OWG has now completed a series of SCSC Newsletter articles to gently explain ontological concepts and notations. The aim is to provide an accessible introduction to the domain so readers can be more informed and to help demystify a topic that may have a perception of being quite academic. The fourth article in the series can be found in this edition.

Lead Dave Banham ontology@scsc.uk

The Safety Futures Initiative

The Safety Futures working group meets once a month to bring together fresh perspectives, innovative ideas and insights.

The Safety Futures is diligently working on developing a comprehensive roadmap to guide new safety professionals in exploring career paths across various industries. To supplement this, they have an upcoming initiative to create a University 'League' Table to rank universities based on their safety engineering courses and the career pathways they support.

They are also looking to start a reverse mentoring programme where we can match more experienced people with new professionals for information exchange.

The group encourages anyone to get involved to help shape the future of safety engineering careers.

Lead Giles Howard giles.howard@scsc.uk

Systems Approach to Safety of the Environment



The Systems Approach to Safety of the Environment Working Group (SASEWG) is a new group intending to apply Systems Safety practices to systems that are embedded within the natural environment, while focussing on that environment.

The group aims to produce clear guidance on how engineered systems should be developed and managed throughout their entire lifecycle so as to preserve, protect and enhance the environment.

Please get in touch with the working group lead if you would like to join or find out more about this group.

Lead **James Inge** james.inge@scsc.uk

Safety Management and Safety Management Systems



This new working group aiming to cover Safety Management and also Safety Management Systems (SMSWG) is now in full swing with almost 60 members.

Safety Management can be a difficult challenge with many different stakeholders with their own objectives pulling in different directions, perhaps one might draw parallels with the equally challenging task of herding cats! This analogy has given rise to the group's icon as shown.

Being an effective Safety Manager requires a balance of technical knowledge, regulatory understanding, soft skills, and leadership. Yet many are asked to lead safety without prior experience or clear support. This group will explore challenges such as overcoming organisational resistance, gaining senior buy-in, building trust across the workforce, and creating a positive safety culture where reporting is safe and encouraged. The group will also explore regulatory frameworks, going beyond compliance to fully understand the intent behind the rules.

The group is led by Si Hays so get in touch with him if you want to find out more about the group.

Lead **Si Hays** si.hays@scsc.uk

SCSC Membership

The SCSC provides a range of services to the System Safety community including seminars, tutorials, leadership events, specialist topic working groups, the annual symposium and a comprehensive body of publications. Membership brings many valuable benefits such as free access to online events and access to papers presentations and other resources from events.

See scsc.uk/membership for more details.

	Public	Registered	Individual	Corporate
Access to the SCSC Newsletter	✓	✓	✓	✓
Access to the E-Journal	✓	✓	✓	✓
Receive monthly e-flyer	×	✓	✓	✓
Attend SCSC 1-day Seminars	×	×	✓	✓
Attend the annual Symposium	×	×	✓	✓
Access to Proceedings of the Symposium	×	×	✓	✓
Participate in SCSC Working Groups	×	×	✓	✓
Access to publications	×	×	✓	✓
Access to video recordings and presentations	×	×	✓	✓
Reduced membership costs	×	×	×	✓
Free full-page spread in SCSC Newsletter	×	×	×	✓
New members announced on website	×	×	×	✓
Share promotional material at 1-day events	×	×	×	✓
Sponsored content in SCSC monthly e-flyer	×	×	×	✓
Company logo on footer of SCSC website	×	×	×	✓

Individual Membership

Individual Membership is essential for engaging with the SCSC, and in particular, attending Seminars, Symposia and other events run by the club. It also gives access to over three decades of papers and presentations from some of the leading experts in system safety.

To become an individual member of the SCSC please register on the SCSC website using the Home->Register menu option. Complete and save your account registration and then verify your email address. Once registered and logged in, select the Membership menu option and pick the membership package you would like to sign up for.

Individual membership can be paid online using a credit/debit card through our secure payment partner Realex Global Payments or contact Alex King for other payment methods. For student or retired member rates please contact Alex King to get your account status changed.

Corporate Membership

Corporate membership affords all the benefits of individual members at discounted rates, plus some other great benefits for promoting your company. These are:

- A free full-page spread in the SCSC Newsletter, which goes out to thousands of system safety practitioners worldwide
- The ability to share promotional material at SCSC 1-day Seminars
- Sponsored content in the SCSC's monthly e-Flyer that goes to everyone registered on the SCSC website
- Your company's logo on the footer of the SCSC website

Your company contact with the SCSC should arrange the membership and any renewals for your organisation. To join as a member covered by a corporate membership, register as per the instructions for an individual member and then contact Alex King to confirm your affiliation.

Renewing Membership

You should be notified by email when your membership is almost expired or shortly after it has expired. These notifications will contain a link to the online renewal page or you will be able to renew when logging onto the website through the 'click to renew' link.

Membership Fees

The following fees are applicable from January 2026 for new and renewing members:

- 1 year Individual Membership: £159
- 2 year Membership: 7% discount: £295
- 3 year Membership: 17% discount: £395
- 1 year SFI Membership: FREE for first year, £35 for years 2 & 3
- 1 year Membership, retired member rate: £35
- For Corporate Membership discounts contact Alex King

Contact Alex King using office@scsc.uk

The SCSC Steering Group

Current Members


Stephen Bull
stephen.bull@scsc.uk


Beatriz Coutinho
beatriz coutinho@scsc.uk


Dai Davis
dai.davis@scsc.uk


Paul Hampton
paul.hampton@scsc.uk


James Inge
james.inge@scsc.uk


Graham Jolliffe
graham.jolliffe@scsc.uk


Kate McDougall
kate.mcdougall@scsc.uk


Yvonne Oakshott
yvonne.oakshott@scsc.uk


Mike Parsons
mike.parsons@scsc.uk


Rudi Redford-Brown
rudi.redford-brown@scsc.uk


John Spriggs
john.spriggs@scsc.uk


Carmen Noll
carmen.noll@scsc.uk


Dewi Daniels
dewi.daniels@scsc.uk


Jane Fenn
jane.fenn@scsc.uk


Giles Howard
giles.howard@scsc.uk


Brian Jepson
brian.jepson@scsc.uk


Alex King
alex.king@scsc.uk


Mark Nicholson
mark.nicholson@scsc.uk


Wendy Owen
wendy.owen@scsc.uk


Davy Pissoort
davy.pissoort@scsc.uk


Karin Rudolph
karin.rudolph@scsc.uk

Note that honorary SCSC membership has replaced honorary Steering Group membership.

Club Positions

The current holders of club positions are as follows:

Managing Director



Mike Parsons 2019-

Steering Group Chair



Dewi Daniels 2024-

Programme & Events Coordinator



Mike Parsons 2014-

Manager



Alex King 2019-

Newsletter Editor



Paul Hampton 2019-

University of York Coordinator



Mark Nicholson 2019-

Honorary Solicitor



Dai Davis 2022-

Website Editor



Paul Hampton 2025-

eJournal Editor



John Spriggs 2021-

Administrator



Alex King 2016-

Diversity, Equity and Inclusion (DE&I) Lead

TBC

Safety Futures Initiative Leads



Beatriz Couthino 2026-

Calendar

October '26

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

November '26

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

December '26

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

January '27

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

February '27

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

March '27

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

April '27

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

May '27

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

June '27

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

July '27

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

August '27

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

September '27

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Events Diary



29 Oct 2026
SCSC Seminar
AI in a Safety Workflow

Hilton London Euston, London, UK

scsc.uk/events-diary/scsc-seminars/ai-in-a-safety-workflow

12 Nov 2026
SCSC Seminar
Safety Assurance for AI Systems

Hilton London Euston, London, UK

scsc.uk/events-diary/scsc-seminars/safety-assurance-for-ai-systems

18-20 Nov 2026
Course
Accident investigation program

London Heathrow Marriott Hotel

novellus.solutions/mec-events/accidentinvestigationlondon-459

3 Dec 2026
SCSC Seminar
New Techniques for Hazard Identification

Hilton London Euston, London, UK

scsc.uk/events-diary/scsc-seminars/hazard-identification-techniques

9-11 Feb 2027
SCSC Symposium
35th Safety Critical Systems Symposium (SSS'27)

Bristol Harbour Hotel in Bristol, UK

scsc.uk/events-diary/safety-critical-systems-club-symposia/safety-critical-systems-symposium-2027


SCSC
Working For The Safe Use Of Systems
Safety-Critical Systems Club (SCSC)
Corporate Membership
Membership for your organisation
Support your safety culture
Network with senior managers
Share knowledge and experience
Develop your staff
Learn about current standards
Understand best practices
Influence new guidance
Raise your organisation's profile
To take advantage of the corporate membership rates, contact office@scsc.uk
www.thescsc.org
www.scsc.uk