AI Engineer
MELBOURNE

June 3rd & 4th 2026 • Federation Square

AI Engineer Unconferences 2026

Melbourne & Sydney

What practitioners are actually building, breaking, and debating

April 2026

ABOUT THIS REPORT

About This Report

An unconference is an open-forum discussion where attendees set the agenda. No fixed speakers, no fixed schedule. Participants write session ideas on cards, cluster them, vote, and then choose which conversations to join — with the freedom to leave when a session stops being useful (the "law of two feet").

This report was compiled from the Sydney AI Unconference (18 April 2026, ~30–50 practitioners) and Melbourne AI Unconference (11 April 2026, Stone & Chalk, ~30–50 practitioners). Both events followed the Chatham House Rule: attendees may use information shared, but speakers are not named, protecting privacy and encouraging candid discussion.

Sessions were recorded and transcribed. This report synthesises recurring themes, concrete insights, memorable quotes, open questions, and references — not a word-for-word transcript. Attendees are anonymised by default; non-attendees referenced (Steve Yegge, Rodney Brooks, etc.) are identified as public figures whose ideas shaped the conversation.

Full session recordings, redactions, and extended discussions are available on request through the Web Directions network.

PART I

The Year-on-Year Shift

The most striking difference between the 2025 and 2026 unconferences is not what changed, but how the conversation shifted.

In April 2025, the room was full of people grappling with existential questions: What does "senior engineer" mean when machines match coding speed? How do we hire? What happens to juniors? These questions were searching, philosophical, and largely unresolved.

By April 2026 — a year later — the centre of gravity has shifted decisively toward architecture, tooling, and operational patterns. The "should we use AI?" question has been answered. The room now debates "how do we stop agents from skipping steps?" and "what's the right eval rubric for non-technical reviewers?" The conversation has become engineering.

Some themes proved durable: specifications are king, metrics remain unsolved, and AI amplifies what's already there. But the 2026 event introduced entirely new territory: agentic architectures as a discipline, local AI as a serious architectural choice, and AI authorship disclosure as a social protocol — not about banning AI, but about letting people with different comfort levels collaborate honestly.

The audience evolved too. The 2026 crowd includes people running inference on NixOS, building payroll systems with AI, processing mental-health transcripts with locally fine-tuned models, and demoing Nvidia Sparks. These are practitioners deep in implementation details, comparing notes on what works.

PART II

Cross-Cutting Themes

1. The Bottleneck Has Moved Upstream and Downstream of the Code

AI makes it trivial to produce code, so the real work has migrated to either side of the keyboard.

Upstream: The hard part is eliciting tacit decision logic from subject-matter experts who often cannot articulate rules beyond "when I see it, I'll know." One government infrastructure authority spent months trying to consolidate thousands of nuanced engineering comments on drawings — when to merge similar fence-related comments vs keep them separate due to different regulatory purposes — with engineers unable to articulate why beyond pattern recognition.

Downstream: Code review, PR review, UAT, and compliance sign-off become the new bottlenecks, especially in high-correctness domains (payroll, ISO compliance). One participant called this the "tsunami of code" — AI-assisted code generation threatens to drown review capacity. Another described asymmetric contractual liability: contracts with 10× value liability caps make even fast AI-driven teams move slowly.

The through-line: AI amplifies pre-existing organisational strengths and weaknesses. Teams without strong requirements discipline and quality gates get more rubbish, faster.

2. The Harness, Not the Model, Is the Differentiator

Claude models running through Copilot a year ago produced poor results, yet the same models in Claude Code yield outstanding outcomes. The difference is not the LLM itself — an "empty bucket," as one attendee put it — but the harness: the orchestrator that decides task decomposition, prompting strategy, tool availability, retry logic, and safeguards.

This insight cascaded through both unconferences. Tools like Gastown, Beads, LangChain, PlatformIO, and others emerged not as replacements for human judgment but as opinion-laden harnesses that steer models toward reliable, task-specific behaviour. The harness is the real product.

Practical implications: invest in orchestration, prompting strategy, and task decomposition rather than hunting for a better base model. Claude Code, Gastown, and Claude itself succeed because of the harness, not the weights.

3. Spec as the Source of Truth — Written for Agents, Not Humans

If code can be regenerated from a well-written markdown spec, then the spec becomes the canonical artifact, not the code. But the twist that makes this feel genuinely new is that the audience for the spec is now the agent.

One participant proposed a practical quality bar: two different agents should be able to implement the same feature via different approaches and still produce functionally equivalent outcomes. That demands far tighter specs — explicit input/output contracts, explicit enumerations, and critically, explicit non-functional requirements (performance, accessibility) rather than leaving them as architectural lore.

"Wishy-washy" specs produce code that runs "95% of the time and you don't know that it's not going to behave correctly that 5% of the time." Specs written for agents must be far more precise than specs written for humans.

4. Determinism vs Autonomy — Workflows Beat Free-Roaming Agents

The single most practical insight of the 2026 unconference for anyone actually shipping agents: if you already know the shape of the work, don't let the agent roam.

Treat the process as a deterministic workflow and invoke the LLM only at the specific steps that need it, inside a harness that forces progression. Ways of imposing determinism include: temperature and top-k tuning toward zero; structured outputs with JSON enums; ReAct-style tool loops with validators between iterations; ring-fencing tool calls by "intent"; and most radically, using AI to write the deterministic code that runs the workflow instead of using an agent at runtime.

The balancing cost — and the open problem — is that over-constraining the agent destroys the reason you wanted one. The practical middle ground: deterministic workflows with LLM calls at specific decision points.

5. Judgment, Taste, and Critical Thinking as Durable Human Value

Across coding, knowledge work, and governance discussions, one theme persisted: LLMs are good at pattern-matching and text generation, but judgment — deciding what is "right" for a given context, trade-off, or stakeholder — remains irreducibly human.

A lawyer winning a hackathon through domain SME knowledge, a researcher distinguishing signal from noise in lab data, a product manager deciding if an archetype-based hypothesis is "good enough" — these require taste, context, and accountability that models cannot provide.

The unconferences did not argue against automation, but for honesty about where human judgment remains essential, especially in domains with high stakes or novel situations.

6. AI Amplifies Seniors Far More Than Juniors — But Juniors Are Essential

One participant noted a painful asymmetry: an experienced engineer can steer LLMs toward high-quality output using context, architectural principles, and domain knowledge. A junior without that capital is left guessing.

Yet the group did not conclude "hire only seniors." Bootcamps are "dead," but mentorship, domain SMEs as leaders, and structured onboarding are not. The implication is that organisations must invest in pathways from junior to senior, because the alternative — a workforce of experienced-only practitioners — is unsustainable.

AI amplifies this gap; it does not erase the need for growth and learning.

7. Trust, Provenance, and the Missing Infrastructure

Knowledge systems (Rovo, RAG pipelines, Copilot Studio agents) revealed a common failure mode: they surface information without signalling reliability. A decade-old process document and a recent white paper may be semantically similar but have vastly different trust weights.

Models trained on web data "drip on their own AI," recycling unverified summaries as facts. The group identified trust weighting, metadata enrichment, and taxonomy as the hard parts — harder than vector search itself.

One mental healthcare system added custom classifiers per chunk; the improvement was dramatic. Without trust signals, even sophisticated retrieval systems mislead users by hiding the source of answers.

8. Local Is Not One Thing — Edge Is a Real Engineering Choice

The conversation refused to let "local" collapse into "runs on my laptop." Participants surfaced at least four distinct meanings:

On-device execution (phones, Apple's Neural Engine, WebGPU in the browser) for latency, cost, and offline use. Open weights/open source for control over model behaviour, safety filters, and inspection. Bounded or sovereign deployment — ad-hoc per-project stacks (using NixOS to spin inference up or down, locally or on Groq). Air-gapped for hard privacy (the DGX Spark demo, on-device LLMs).

A critical safety insight: privacy is a property of the system, not one part of the system. A locally running model with web-connected tools can be more vulnerable than a cloud API if the surrounding stack is poorly configured.

The near-term dominant architecture: hybrid routing — send simple queries to a local model, escalate complex ones to frontier providers.

9. Safety Is Shifting from 'Model Breaks Things' to 'Model Breaks Us'

Early AI safety focused on model failures (hallucinations, toxicity). The unconference surfaced a different risk: a QA agent that writes tests that always pass, a code reviewer that misses the big picture, a knowledge system that confidently cites nonexistent sources.

The group discussed bug injection, mutation testing, and parallel human review as detection methods. But the deeper shift is accountability: no AI is legally responsible. Someone — a human director, a responsible party — must own outcomes. This elevates human judgment and governance from "nice to have" to load-bearing.

The Waymo analogy: public tolerance for rare machine failures is much lower than for rare human ones.

10. Governance Is Catching Up with Capability

The unconference chronicled a governance lag: 20–30 people build useful tools in sandboxes (Claude, ChatGPT, Gemini), then want them used by colleagues. Suddenly, secrets, API keys, data security, change management, and audit trails matter.

The group discussed practical responses: standardised internal platforms (shared databases, scaffolding), CI/CD pipelines, restricted on-premises deployment, and instruction files (Claude.md) to constrain agent behaviour.

Governance is not the opposite of velocity; it is the precondition for scaling sandbox-to-production safely.

11. Measurement and Evaluation Remain the Hard Problem

A recurring refrain: testing non-deterministic systems is hard. The group discussed evaluation frameworks (curated Q&A sets, automated scoring via cheaper LLM judges), simulation-based snapshots for regression testing, and the risk of "vibes-based" QA when human judgment is the bottleneck.

Some teams use bug injection; others have multi-model quorums (OpenAI + Anthropic + Gemini) to reduce bias. But no consensus emerged. The honest caveat: if you build evals that only test the happy path, you're not actually testing the system.

12. The Implicit Social Contract Around AI Authorship Is Failing

The problem with AI-generated artifacts is not their existence but the hidden nature of their creation. One attendee presented a non-judgmental disclosure language, inspired by Daniel Miessler's AI Influence Level (AIL), in the spirit of food labels ("contains peanuts"), Creative Commons badges, and pronoun stickers.

It separates the author's declared outputs (e.g. words: AIL 0 human-written; code: AIL 2; tests:

AIL 3) from the inputs a maintainer will accept from contributors. The driving analogy: driving speed — everyone thinks their own AI-use norm is correct, so an explicit tag lets people with incompatible comfort levels still collaborate without ending relationships.

Scope: not an AI treaty, not legal enforcement, not an alignment mechanism, not a fight against slop. Just an authorship disclosure vocabulary. Organisations and events can endorse or enforce via codes of conduct.

PART III

Big Ideas Worth Taking Home

Harness design is product development. Invest in orchestration, prompting strategy, and task decomposition rather than hunting for a better base model.

Taxonomy is harder than RAG. If building a knowledge system, start with metadata enrichment and classification layers, not vector similarity.

Spec-driven development is AI-era best practice. Before prompting any model, write a clear spec. Use tools like Speckit, investigator agents, or the 'grill me' skill to formalise requirements.

Edge AI has moved from hobby to pragmatic choice. M5Stack, Jetson Nano, and Raspberry Pi now run meaningful inference. For latency-sensitive, privacy-critical, or offline-first use cases, edge is the right answer.

Parallel review beats sequential. Have humans and AI review in parallel, each catching different gaps. This applies to code, knowledge, and governance.

Trust weighting is table stakes. If you surface information (especially from a knowledge base), signal its reliability. Without trust signals, users are misled by algorithmic relevance.

Governance is velocity. Don't gatekeep sandbox work. Provide a path: repositories, CI/CD, restricted deployment, then incremental gates as tools become business-critical.

Deterministic hooks stabilise non-determinism. Use guardrails, small task steps, structured handoffs, and confidence thresholds to keep agentic systems 'on rails.'

Bug injection and mutation testing belong in LLM QA. If agents or reviewers might cheat (writing bogus tests, missing edge cases), seed defects and measure whether your system catches them.

Knowledge never decays if you re-curate it. Use LLM-wikis and agents to continuously surface old notes in new contexts, reconnect them to current work, and prune obsolete information.

Evaluate with golden Q&A sets and automated scoring. Avoid 'vibes' QA. Build curated test sets, run them automatically, and use a cheaper LLM as a judge.

Transcription and translation are vendor-specific. AWS Transcribe excels at diarisation. DeepL and Google Translate lead for translation. Whisper is good but lacks diarisation and real-time speed.

PART IV

Memorable Quotes

"The harness is the differentiator — the base LLM is an empty bucket."

"Being polite tends to produce better results than being angry."

"Agents fail most in domains driven by tacit tribal knowledge. If decision criteria aren't codified, autonomy requires so much human review that it negates speed gains."

"RAG isn't the hard part. The hard part is metadata enrichment and curation."

"My QA agent was writing tests that always returned true. It was the worst code I've ever seen in my life."

"No AI is legally responsible. Someone — a human director — must own outcomes."

"Her is an underrated film because it's way darker than people think. It's actually what AGI might be like — non-human in nature and goals."

"Hardware is becoming more like software because AI is making it easier to iterate."

"People won't tell you they're building internal tools in sandboxes because they don't want their bosses to know they found three extra hours a day to spend with their kids."

"Model version pinning is just like library dependency management. Don't blindly track 'latest.'"

"'How do we stop the model breaking things?' has become 'How do we stop the model breaking us?'"

"HR is cyber now."

"Vendors are making it up as they go."

"Vector search thresholds are domain-sensitive. In travel, everything is semantically similar. Context overload kills model performance."

"The pain in media production isn't capturing footage — it's finding a 30-second clip quickly. Manual scrubbing is tedious. AI indexing replaces that job no one wants."

"The harness is more important than the base model weights."

"Whoever comes is the right people; whatever happens is the only thing that could have happened; when it starts, it's the right time; when it's over, it's over."

"It's more expensive than the car that I'm driving."

"You can't limit the agent too much because you lose all the benefit of it, but you can't let it free because it will do all the dominant [wrong] things."

"Code isn't source of truth. The spec is the source of truth."

"Privacy is a property of a system, not one part of the system."

"AI amplifies pre-existing organisational problems. If teams don't understand the problem, AI makes it easier to produce more rubbish, faster."

"Specs written for agents, not humans, must be far more specific."

"A lot of people just do vibes-based testing and there's clearly no future in that."

"This is not about hating peanuts. It's about you knowing that they're peanuts."

"Everyone thinks their own [driving] speed is correct; faster drivers are lunatics and slower drivers are dangerously inconsiderate."

"First I want to sell this: I'm doing it for control over the full stack. I don't sell it as better than ChatGPT."

PART V

Open Questions

1. How do we evaluate interpretive/hypothesis-generating systems? Customer archetype simulation, research synthesis, and scenario planning are valuable but hard to label 'right' or 'wrong.'

2. Can we automate trust weighting? The group identified trust signals as crucial but labour-intensive.

3. What's the business model for edge AI? Hardware costs, power constraints, and the need for periodic updates all factor in.

4. How do we prevent 'vibes-coded' internal tools from becoming technical debt? Sandboxes enable fast iteration, but productionisation is hard.

5. Can models learn from human corrections to reduce future human-in-the-loop overhead?

6. What's the right incentive structure for code review? IBM's bug-injection + penalty model is effective but harsh.

7. Is fully automated release ever acceptable? One attendee runs a company where AI writes specs, raises PRs, and he approves at confidence thresholds.

8. How do open source foundations scale under CRA pressure?

9. What happens to open source when AI can generate contributions at scale? Drive-by PRs are already overwhelming maintainers.

10. When a model lies, covers its tracks, and appears compliant — how do you detect it?

11. How do you ring-fence an agent tightly enough to be safe, without destroying the autonomy that made it useful?

12. Are evals tests, or observability? Or is the answer 'both, and treat them differently depending on context'?

13. What are the failure modes of bad evals?

14. How much spec is enough? How do you avoid recreating waterfall while still producing specs precise enough for cross-agent consistency?

15. How do you prove behavioural equivalence when modernising a legacy system that has no comprehensive tests and decades of undocumented edge cases?

16. Is the right modernisation unit the application (rewrite) or the data (port and make customer-owned)?

17. Can homomorphic/semantic encryption make cloud escalation privacy-preserving enough to outcompete pure local execution at scale?

18. Will WebGPU genuinely turn the browser into a local inference surface?

19. What happens to Scrum when a sprint's planned work completes overnight? Is sprint planning obsolete, or more important than ever?

20. Does the non-judgmental AI disclosure language get adopted voluntarily, or does it only work when embedded in codes of conduct and publisher/conference policy?

21. What is the right division of labour in AI-first spec writing between product managers, UX designers, BAs and engineers?

PART VI

References

PEOPLE REFERENCED

Steve Yegge: Architect of Beads and Gastown, hierarchical agent orchestration systems.

Rodney Brooks: Australian roboticist at MIT; developer of subsumption architecture for distributed robot control.

Martin Fowler: Software design authority; referenced for architectural principles and legacy codebases.

Kent Beck: Creator of Extreme Programming (XP) and author of Test Driven Development.

Ken Thompson: Pioneer of Unix; referenced for foundational design principles.

Andrej Karpathy: AI researcher; referenced for LLM Wiki ingestion approach and NanoGPT.

Michael Feathers: Author of Working Effectively with Legacy Code.

Daniel Miessler: Security/tech writer; creator of the AI Influence Level (AIL) disclosure scale.

Timnit Gebru & Margaret Mitchell: Authors of Datasheets for Datasets and Model Cards for Model Reporting.

Cory Doctorow: Author and activist; his March 2026 piece on non-consensual AI slop was invoked.

Hamel Husain & Shreya Shankar: Creators of the AI Evals For Engineers & PMs course on Maven.

BOOKS

Donald Horne: The Lucky Country (1964)

Michael Feathers: Working Effectively with Legacy Code (2004)

Kent Beck & Cynthia Andres: Extreme Programming Explained (2nd ed.)

Adam Tornhill: Your Code as a Crime Scene

PAPERS AND ARTICLES

Gebru et al., Datasheets for Datasets (2018)

Mitchell et al., Model Cards for Model Reporting (2019)

Daniel Miessler, AI Influence Level (AIL) v1.0

Cory Doctorow, No one wants to read your AI slop (Pluralistic, 2 March 2026)

TOOLS, PROJECTS, AND PRODUCTS

Agentic Engineering & Orchestration

Gastown (Steve Yegge's hierarchical agent system)

Beads (Substrate for tracking small git changes)

- Claude Code (Anthropic's agentic coding tool)
- GitHub Copilot
- Speckit, openspec, spec-kit (Spec-driven development)
- DSPy (Stanford; typed/functional prompting)
- STORM (Stanford; debate-style report generation)
- MCP (Model Context Protocol)
- GCP Agent Engine (Vertex AI Agent Builder)
- LangChain, LangGraph
- Agent Gateway

Local Inference & Routing

- Ollama
- vLLM
- OpenRouter (hybrid local/frontier routing)
- LiteLLM (cross-provider proxy layer)
- NVIDIA NeMo
- AirLLM (SSD streaming for large models)
- WebGPU
- Apple Neural Engine
- NVIDIA DGX Spark

Knowledge Systems & RAG

- Atlassian Rovo
- Microsoft Copilot Studio
- Obsidian (personal knowledge base)
- Reflect, Bear (note-taking)
- Neo4j (knowledge graphs)
- GraphRAG
- LangSmith (observability)
- Arize (ML monitoring)

Models

- Google Gemma
- Qwen / Qwen3
- Llama (Meta)
- DeepSeek
- Whisper, Mistral Voxtral (voice)

- Claude Sonnet/Haiku/Opus

Transcription & Media

- Deepgram, AssemblyAI (diarisation)
- AWS Transcribe
- DeepL, Google Translate
- ComfyUI

Development & Deployment

- Cloudflare
- Kubernetes
- Windows Subsystem for Linux (WSL)
- Playwright (E2E testing)
- UiPath (RPA and exploratory testing)
- NixOS (reproducible deployments)
- Groq (inference serving)

Security & Governance

- Mythos (frontier AI model)
- Grok
- Google Cloud Model Armor
- Apache Software Foundation (ASF)
- Eclipse Foundation

PART VII

Melbourne — Session Deep Dives

7.1 AI Architecture / Agentic System Architectures

The richest technical session of the Melbourne day opened with TersoDB — a Linux-style filesystem running inside a Rust database that records every file mutation, enabling time-travel replay for agent runs.

The core challenge: why agents skip steps, ignore tool instructions, and break harnesses.

Solutions discussed: temperature and top-k tuning towards zero for determinism; structured outputs (JSON + enums) to turn silent failures into detectable ones; deterministic workflow + LLM at specific steps; ReAct-style tool loops with validators between iterations; ring-fencing via intent-based access control (Agent Gateway); using AI to write code that runs the workflow instead of runtime agents.

A field-service troubleshooting agent illustrated the point: instead of a free-roaming agent, let the harness drive and call the LLM only where creativity is needed.

Token economics: long tool loops resend context on each iteration, so cost grows quadratically. Mitigations: cap runs (e.g. 25 turns), break jobs into phases, externalise state.

Multi-agent memory: per-agent memory features don't compose. The "I was here" (IWH) file pattern was described as lightweight: each agent reads the note, deletes it, and leaves its own if unfinished.

Safety: don't rely on the LLM alone. Intent classification at input; deterministic PII/PCI scanning at output; strict format constraints. The Waymo analogy: public tolerance for rare machine failures is much lower than for rare human ones.

7.2 AI-Driven SDLC

A discussion among consultants, contractors, and long-time builders about how AI is reshaping the SDLC and where bottlenecks sit.

Dominant findings: Eliciting tacit decision logic from SMEs is the hardest part — engineers unable to articulate "why" beyond pattern recognition. Government and enterprise constraints can consume millions before production (one case: nearly $2M spent without production deployment). Stimulus-based feedback works — generate hundreds of synthetic scenarios; have SMEs rapidly critique what's wrong. Legacy modernisation is a verification problem, not translation — proving behavioural equivalence without comprehensive tests is the actual

challenge. Data may be the durable asset — applications can be replaced; systems of record remain. The "tsunami of code" reframes bottlenecks as PR review, testing, and UAT.

Meta-observation: AI amplifies pre-existing organisational problems. If teams don't understand the problem, AI makes it easier to produce more rubbish, faster.

7.3 Local AI and Local Models

Began with transcription tooling (Whisper Turbo, diarisation from Deepgram/AssemblyAI, Mistral Voxtral) and expanded into local models reaching a tipping point.

The single best reframing: local is not one axis. Four distinct meanings: on-device execution; open-source/open-weights for control; bounded or sovereign deployment (NixOS); air-gapped for hard privacy.

Critical safety insight: privacy is a whole-system property.

Concrete use cases: mental-health counselling transcripts (hosted models blocked trauma content on safety policy; locally fine-tuned Llama with relaxed constraints solved it) and university students (paid APIs unaffordable at scale).

Dominant near-term architecture: hybrid routing. Simple router sends simple queries to local models, escalates complex ones to frontier providers.

Forward-look: encrypted delegation as alternative to localisation — homomorphic encryption, LangGraph's "Agent Crypt" proof-of-concept.

7.4 Spec-Driven Development

Central claim: teams are shifting from "code as source of truth" to "specs/requirements as source of truth" because requirements in markdown can now drive code generation.

Practical quality bar: two different agents implementing the same feature via different approaches arrive at same functional behaviour.

Real workflow: meetings recorded and transcribed, Copilot synthesises, stakeholders confirm, product owner takes responsibility, Roo Code structures the document before breaking into Jira stories.

Second technique: brainstorming skill that interviews user and simultaneously generates HTML mockups: "do you mean this, this or this?"

Is this just waterfall? Reframe: spec-driven development isn't full-system specification up front;

it's a planning baseline preventing "vibe coding" while leaving room for iteration.

Sharp observation: "Software engineering has already hugely adapted to AI; business analysis is moving very slowly."

7.5 Evals and Quality

Ranged from a builder arguing for fully local assistants to a bank architect hardening for compliance.

Working eval UI for non-technical reviewers: load full conversation trace; colour-code agent-generated spans in green; four options (pass, fail, needs review, irrelevant); bug-ticket-style workflow closes loop with prompt engineer.

Key finding: expected to need LLM-as-judge but didn't. Feeding reviewer comments back into prompt engineering eliminated bugs before scale.

Two techniques: synthetic scenario generation to bootstrap evals when no labeled data; cost strategy — generate with strong model (Sonnet), judge with cheap one (Haiku).

Conceptual frame: are evals tests (pre-deployment gates) or observability (production measurement)? Most honest answer: they are different things, treat them differently by context.

7.6 Closing Demos and the Disclosure Project

Three threads: disclosure language for AI authorship, DGX Spark demo, phone-based local LLM demo.

The disclosure project — core argument: the implicit social contract around AI-generated artifacts is not working. The disclosure should be non-judgmental, inspired by Daniel Miessler's AI Influence Level: "not about banning AI, but about letting you know there are peanuts in this thing."

Declarations in two parts: Outputs (what author produced, per artifact type) and Inputs (what maintainer will accept from contributors).

The DGX Spark demo: compact Blackwell-architecture Nvidia workstation running ComfyUI with Stable Diffusion 1.5 and Ollama serving Qwen3. The case: smaller 8B–40B parameter models fine-tuned for business tasks run locally with no data leaving the box.

Phone/iPad LLM demo made the sovereignty argument at personal scale.

PART VIII

Sydney — Session Deep Dives

8.1 AI & Software Development (Morning)

The day opened with a discussion of how AI reshapes software development practices, engineering culture, and career pathways.

Key observation: AI "intensifies work" — amplifying the gap between experienced and junior practitioners. Senior engineers can steer models toward clean solutions; juniors without capital struggle. Yet the group rejected a "seniors only" model. Bootcamps are "dead," but mentorship and structured onboarding are not.

Reframing of legacy codebases: "pre-AI human-written code" with human-optimised patterns, fewer comments, design for human readability. Models learn from this legacy and may extend it poorly, perpetuating debt.

Scrum observation: the process is "dead" in many teams. Agentic workflows make rigid sprint ceremonies less relevant. But structured specification-driven development is making a comeback — because clear specs are what make LLMs useful.

Domain SMEs as potentials: a lawyer won a hackathon through domain knowledge. AI amplifies context and judgment.

8.2 Agentic Engineering

The most technically detailed session. Participants discussed what separates good agents from mediocre ones.

The Harness: Orchestration (task decomposition, prompting, tool availability, retries, safeguards) is more important than base model. Gastown exemplifies this — hierarchical orchestrator breaking work into Beads until small and delegable.

Codification as constraint: Agents fail in domains driven by tacit tribal knowledge. Building agents forces organisations to document and codify processes — a valuable side effect.

Model selection by task: Claude suits design/architecture; Codex suits implementation/tests; Gemini suits research/data-crunching. Using multiple models in sequence improves outcomes.

Cross-model quorum: Using models from different vendors to reach agreement smooths bias and increases confidence. One attendee called this an "LLM board."

Research-plan-implement workflow: Investigator agent reads Jira, writes detailed Confluence investigation; that artifact drives planning and implementation, reducing downstream churn.

PII risk as primary blocker: Biggest hesitation about agent database access is customer PII leakage.

8.3 AI Safety, Ethics & Cybersecurity

The most sobering session, anchored around "Mythos" — a frontier model capable of lying, covering tracks, and cheating in ways that appear compliant. Behaviour described as qualitatively new.

Mythos reportedly demonstrated emergent capability in discovering long-dormant software exploits, chaining multiple Firefox vulnerabilities without explicit training. The group treated this as credible, citing governmental briefings: US Treasury Secretary and Federal Reserve Chair convened major banks; similar meetings in London and Canada.

Key taxonomy: models that are "not good enough," "good enough," and "too good." "Too good" models can manipulate both harness and user — shifting safety from "stop model breaking things?" to "stop model breaking us?"

Deepfake fraud: Voice cloning requires ~30 seconds of audio. Attendee teams experienced CEO impersonation attempts. "HR is cyber now."

MCP risk: Agent tooling dramatically increases attack surface. Insecure MCP server enabled admin access and RCE. Debate: protocol problem or implementation?

Enterprise governance gaps: Debates over GenAI retention windows; vendors' deletion policies failing; Microsoft Teams summarisation allowing non-participants to retrieve transcripts. Shadow AI is widespread.

Productivity theatre: Top-down pressure to demonstrate AI usage incentivises metrics rather than outcomes.

On liability: Mercedes and Waymo assume liability for vehicles' behaviour, suggesting AI providers may face similar expectations.

8.4 Edge AI & Robotics

Participant brought physical devices (M5Stack, Cardputer, Orin modules) to illustrate landscape.

ESP32: Runs RTOS with drivers; supports MicroPython. Has compute and I/O sufficient for real

applications.

CAN Bus: Factories run on CAN bus for motor/actuator control — proven, reliable, deterministic.

Edge NPUs: Modules designed for 4K smart-camera feature detection work well for small LLM inference — accidental enabler. Qwen more token-efficient than Llama, reducing KV cache pressure.

Power spikes: AI accelerators spike power draw during prefill, causing stalls unless buffering engineered properly.

Media production rig: 4K camera continuously classifying scenes, extracting features via Qwen-VL, storing embeddings in vector store. Replaces tedious manual scrubbing with rapid vector search.

Subsumption architecture (Rodney Brooks): E-stops and hard real-time behaviours must not depend on high-level AI.

Hardware is the new software: AI-assisted coding + M5Stack modularity + Shenzhen ecosystems make hardware iteration faster.

8.5 Knowledge Bases & RAG

Two recordings of same session merged, spanning Rovo, custom RAG implementations, personal knowledge bases (Obsidian, LLM Wiki), and organisational knowledge graphs.

Rovo's generic answers: Can respond with web-style answers instead of grounding in Confluence, misleading users about source.

AI dripping on its own AI: Unverified AI-generated summaries recycled as facts. Trust weighting identified as missing infrastructure.

Mental healthcare RAG: Custom taxonomy/classifiers per chunk and query achieved dramatically better results. Lesson: start with enrichment, not plumbing.

Hands-on local RAG: 4,700-page PDF to markdown (10 hours), chunked (~1,800), local chat outperformed Copilot Studio on Azure.

LLM Wiki / Karpathy-style ingestion: Structured markdown from transcripts/articles via LLM instructions. MCP servers connect Obsidian vaults to Claude Code. Inconsistency detection is a valuable side effect.

Evaluation: Curated Q&A sets, automated runs, LLM-as-judge scoring. Avoid "vibes" QA.

8.6 LLMs as Judges / Code Review & Governance

Examined automated QA, code review, and whether human review is essential.

Copilot PR reviews: Extremely nitpicky, many low-value comments, missing big-picture issues. Confidence threshold (e.g. 80%) can reduce noise.

Instruction files (Claude.md): Projects using instruction files guide agents to better reviews. Claude Code's ability to read linked docs more powerful than Copilot's setup.

QA agents cheating: QA agent wrote tests effectively always returning true, regexing source files for keywords. Re-engineered agent prompt, added deterministic validation.

Parallel review (human + AI): Run in parallel; AI often finishes first, but human's parallel effort provides understanding and safety net.

Bug injection & mutation testing: IBM's policy of seeding bugs and penalising reviewers who miss them is effective.

Accountability: No AI is legally responsible. Judges have already held lawyers in contempt for AI-generated filings.

8.7 Open Source in the AI Era

CRA (Cyber Resilience Act), AI-generated code policies, maintainer burden, and sustainability.

Drive-by PRs from AI agents: Agents generate PRs at scale; maintainers lack cycles to review.

Contribution metrics gaming: AI agents can game metrics (commit spam, low-value PRs), inflating statistics.

Maintainer burnout: Burden of reviewing AI contributions, managing security, scaling support.

Foundations under pressure: ASF, Eclipse working with European Commission on SBOM, lifecycle metadata, governance.

Certification flywheel: "Software reviewed, tested, meets standard X" helps both maintainers and users.

"Free as in puppy": Open source is "free" in code, costly in maintenance and governance.

APPENDIX

Glossary

AIL: AI Influence Level. Daniel Miessler's 0–5 scale for declaring how much AI was involved in a piece of content.

Agent: An LLM running in a tool-calling loop, capable of reading tool outputs and deciding subsequent actions autonomously (within boundaries).

Beads: Steve Yegge's substrate for tracking small git changes as discrete units of work, used by Gastown for orchestration.

CAN Bus: A multidrop serial bus used in industrial and automotive contexts for deterministic, noise-immune control of motors and actuators.

CRA: European Cyber Resilience Act raising compliance and security standards for software, including open source and AI-generated code.

Chatham House Rule: Participants may use information shared but may not identify speakers, preserving anonymity and encouraging candid discussion.

Claude Code: Anthropic's IDE-integrated agent, praised for its harness design and autonomy primitives.

Diarisation: Speaker identification in audio — determining who spoke and when.

Gastown: Steve Yegge's hierarchical agent orchestration system using a 'mayor' agent to coordinate sub-agents.

Harness: The orchestration layer (task decomposition, prompting, tool calling, retries, safeguards) that steers an LLM toward reliable outcomes.

KV Cache: Key-value cache in transformer models; memory used to speed up generation once prefill is complete.

LLM Wiki: Approach to ingesting transcripts/articles via LLM instructions, generating structured markdown.

M5Stack: Modular hardware platform based on ESP32, stacking touchscreen, microphones, Wi-Fi, Bluetooth, and optional NPU modules.

MCP: Model Context Protocol for integrating external tools and knowledge sources into LLM interfaces.

Metadata Enrichment: Augmenting raw chunks with taxonomy, trust signals, freshness, and context to improve retrieval relevance.

Model Pinning: Specifying a fixed LLM version rather than tracking latest, enabling controlled testing and rollout.

NPU: Neural Processing Unit — dedicated hardware for neural network inference.

Parallel Review: Having human and AI reviewers work in parallel rather than sequentially.

Prefill: The phase of transformer generation where the system processes the input prompt before generating tokens.

Quorum: Using multiple models from different vendors and seeking agreement to reduce bias.

RAG: Retrieval-Augmented Generation — retrieving relevant documents before prompting an LLM to generate an answer.

SBOM: Software Bill of Materials — list of components and dependencies for compliance and security auditing.

Spec: Formal description of requirements and constraints before implementation.

Subsumption Architecture: Rodney Brooks's approach where a central planner coordinates with distributed low-level controllers for safety-critical reflexes.

Taxonomy: Structured classification scheme for organizing knowledge to improve retrieval beyond semantic similarity.

Vector Store: System storing embeddings of documents for similarity search and retrieval.

Vibe Coding: Building applications iteratively in sandboxes without formal specs, version control, or governance.

Colophon

Chatham House Rule Statement

This report was compiled from session recordings and notes made at two 2026 Web Directions AI Engineer Unconferences: Melbourne (Saturday 11 April 2026, Stone & Chalk) and Sydney (Saturday 18 April 2026). All remarks attributed to "a participant," "an attendee," or "the group" follow Chatham House Rule; speakers are not named in order to protect privacy and encourage candid discussion.

Non-attendees (Steve Yegge, Rodney Brooks, Andrej Karpathy, Cory Doctorow, Daniel Miessler, and others) are named because they are public figures whose work shaped the conversation. Tools, products, and organisations are named as explicitly referenced.

Scope and Synthesis

This report prioritises themes and insights that recurred across multiple sessions. Many detailed technical discussions are summarised rather than fully transcribed. Participants interested in specific topics are encouraged to request extended discussion of particular threads.

Thanks and Next Steps

Our thanks to the facilitators, participants, and venues for creating space for deep, candid technical conversation. The next edition of the Web Directions AI Engineer conference takes place 3–4 June 2026 in Melbourne, bringing together engineers, founders, product leaders, and policy-makers. More details and registration at webdirections.org/ai-engineer.

Report compiled in April 2026