# Document Suite Map

Reading Paths for *Semantic Defects in AI-Generated Code* and Companion Documents

Discussion Paper — Draft for Comment v0.1.0


| | |
|---|---|
| **Date:** | **24 March 2026** |
| **Prepared by:** | John Morrissey |
| **Parent paper:** | Semantic Defects in AI-Generated Code: Assurance Frameworks for AI-Assisted Development in High-Stakes Code Paths (v0.1.0) |

## What this is about

AI coding tools are in active use across organisations and their contracted suppliers. These tools are productive and increasingly standard — this body of work does not recommend restricting them. It identifies a specific class of defect they produce that existing assurance frameworks were not designed to catch: code that is syntactically correct, passes all automated checks, looks right to reviewers, but makes the wrong decision about data that matters in its institutional context — a silently defaulted security classification, a swallowed audit record, an unvalidated external authority claim treated as trusted.

These documents present one argument at different depths. They describe the problem, classify the failure modes using a STRIDE-based taxonomy, and assess where existing guidance (including the ISM and Essential Eight) leaves gaps. The work is informed by measured defect data from a real project.

## How the documents fit together

*Governing AI-Generated Code* (13 pp) is the entry point for all audiences — it establishes the problem and summarises the response.

*Reviewing AI-Generated Code: A Practical Guide* gives hands-on review guidance for staff who use AI to write code but do not have CI pipelines or developer tooling.

The *Discussion Paper* (*Semantic Defects in AI-Generated Code*) is the full technical analysis underpinning both. It includes the complete STRIDE-based failure taxonomy, worked case studies, an annotated agent transcript, cross-model defect chaining analysis, and a systems thinking primer. It is detailed — approximately 200 pages — and is available for readers who need the evidence base.

## Reading paths by role

### Leadership, policy, and non-developer staff

| Role | Start here | Then if needed |
|---|---|---|
| **Executive leadership** | Governing AI-Generated Code (13 pp) | Discussion paper Exec Summary (2 pp), App E §E.7 (1 pp cross-cutting observations) |
| **Policy officer / adviser** | Governing AI-Generated Code | Discussion paper Exec Summary, §6 (guidance gap analysis), App E §E.4–E.7 |
| **Non-developer staff using AI** (analysts, data engineers, ops staff) | Practical Guide (23 pp) ⟶ Governing AI-Generated Code (for context) | For SQL/data work: Discussion paper App C (SQL extension). For worked examples: App D (simulation), App E (transcripts) |

## Security, assessment, and assurance

| Role | Start here | Then if needed |
|---|---|---|
| **CISO / Security adviser** | Governing AI-Generated Code → Discussion paper §6 (gap analysis) | Discussion paper §2, App E §E.7 (delegate to security architecture team if needed) |
| **CTO / CIO** | Governing AI-Generated Code → Discussion paper §1–2, §6 | Discussion paper §8 (case study), App D (simulation), App E (transcripts) |
| **IRAP assessor** | Discussion paper §7 (response landscape), §9 (open questions — evidence thresholds), App A | Discussion paper §6 (gap analysis), App D (simulation), App E (transcripts) |
| **Auditor / assurance** | Discussion paper Exec Summary, App E §E.7 | Discussion paper §9 (open questions — governance mechanics, evidence thresholds), App E §E.4–E.5 (policy-not-applied pattern) |
| **Programme director** | Governing AI-Generated Code → Discussion paper §1.2.6, §9.6, App E §E.7, §E.6 (spec-level review — catching violations before code is written) | Practical Guide (for team distribution) |
| **Procurement / contracts** | Governing AI-Generated Code §5–6 (the contracted development dimension) | Discussion paper §6.7 (contracted development), App E §E.4–E.5 |

## Development and implementation

| Role | Start here | Then if needed |
|---|---|---|
| **Developer** | Practical Guide (23 pp) → Governing AI-Generated Code (for context) | For senior developers: Discussion paper §2, §4, App A, App D (simulation), App E (transcripts) |
| **Development team lead** | Governing AI-Generated Code → Discussion paper §2, §4, §7, App A, App D (simulation), App E (transcripts) | Discussion paper §7.2 (technical controls — what is buildable) |
| **Tool implementer** | Discussion paper §2–3, App A (detection approaches per entry) | Discussion paper §7.2 (technical controls), §8 (case studies), App E (transcripts) |

# The documents

| Document | Pages | Audience | What it covers |
|---|---|---|---|
| **Governing AI-Generated Code** | ~13 | Everyone | The problem statement: what the defects are, why they are not targeted by existing checks, why updated assurance is needed. Entry point for all other reading. |
| **Practical Guide** (Reviewing AI-Generated Code) | ~23 | Staff using AI to write code | Five review questions, worked code examples, hot-path identification — for people copying code from AI |

| Document | Pages | Audience | What it covers |
|---|---|---|---|
| | | | chat windows without CI or developer tooling. |
| **Discussion Paper** (Semantic Defects in AI-Generated Code) | ~200 | Technical leads, assessors, security architects | Full analysis: 20-entry failure taxonomy (ACF), STRIDE mapping, two case studies (a simulation with full source code and a longitudinal observation with annotated transcripts), cross-model defect chaining, systems thinking primer. v0.1.0. |