The AI-Augmented SDLC

This is the complete lifecycle described in this book, condensed to a single reference. Pin it to the wall.

1. INTENT

Sequence: idea → iteration → clarification → specification

Start with a rough idea. Prototype fast — throwaway approaches, wrong answers discarded in minutes. The triad (product, engineering, quality) iterates live. Clarification emerges through iteration: you discover what you need by seeing what works and what does not. The spec captures what survived — the durable artifact, the audit trail, the scope constraint.

The spec is the output of the thinking, not the gate at the entrance.

Five forcing-function questions (minimum bar for Tier 2+ features):

1. What problem does this solve?
2. How will we know it worked?
3. What is out of scope?
4. What must NOT happen?
5. Pre-mortem: why could this fail?

Spec depth scales with risk tier. A config change gets a one-line intent note. A new API endpoint gets a one-page spec. A multi-service feature gets edge cases, constraints, and a rollback plan.

2. GENERATION

AI builds from the spec, scoped by risk tier.

- **Implementation phases:** 400 lines each, independently verifiable, independently committable
- **Fresh context per phase:** each phase gets full context budget, reads the codebase as it is now
- **Human presence during generation:** the triad is in the room — course-correcting in minutes, not days

3. VERIFICATION

Four fates of a defect:

1. Discarded before entering the pipeline (cheapest — throwaway prototypes)
2. Machine catches it (gates)
3. Human catches it (review)
4. Escapes to users (production escape)

Gate tiers (0-5):

Tier	What it catches	Examples
0 – Static Analysis	Syntax, types, secrets, formatting	Linting, type checking, secret detection, PR size limits
1 – Contract Gates	API shape drift	Schema validation, interface checks
2 – Invariant Gates	Business rule violations	Idempotency, uniqueness, balance consistency
3 – Policy Gates	Security and compliance	No PII in logs, auth on every endpoint, dependency verification
4 – Behavioral Gates	Runtime drift	Feature flags, circuit breakers, progressive rollout, baseline monitoring
5 – Human Review	Judgment calls	Intent alignment, architecture decisions, gate quality inspection

LLM-as-judge operates across tiers — filters what is worth human attention. Structured prompts on economical models beat unstructured prompts on expensive ones.

4. REVIEW

Human judgment operates at two distinct points:

During generation — the right people in the room while code is being written. Catches misalignment before it enters the pipeline. This is the triad model: product, engineering, quality perspectives iterating live.

After generation — humans verify that quality gates are working and handle judgment calls that mechanical evaluation cannot. Humans inspect gate rules, review flagged issues, and periodically check what the gates did *not* flag.

Weekly review: 15 minutes. EM, tech lead, one rotating IC. Read the scorecard, autopsy one outlier, pick one control tweak. The agile retrospective, compressed and sharpened.

5. GUARDRAILS

Scope constraints:

- PR size limits enforced in CI (reject over 400 lines)

- Spec link required for Tier 2+ changes
- Agent sandbox: no production access, no network except allowlist, scoped permissions

Agent boundaries:

- Documented scope per agent (what it can and cannot do)
- Exit conditions: max iterations, timeout, token budget
- Blast radius assessment: what is the worst this agent can do?
- Human override: kill switch tested monthly
- Audit trail: every tool call, decision, and outcome logged

Safety infrastructure:

- Behavioral monitoring with baselines (alert on drift, not spot-check)
- Circuit breakers: auto-revert to human-in-the-loop when metrics degrade
- LLM-as-judge as continuous filter, not replacement for human review
- Rollback plan for every feature: trigger signal, method, owner

The rule: constraints enable speed. Organizations with comprehensive governance adopt AI faster than those without it.

---

6. MEASUREMENT

Verification Triangle — three vertices, tracked weekly:

Vertex	Headline metrics
Intent Clarity	First-pass acceptance rate, post-merge rework rate, increment frequency
Verification Quality	Machine catch rate, change failure rate
Cost	Cost per accepted change, lead time to accepted change

What "closing the gap" looks like: machine catch rate rising (gates absorb more), human save rate falling (because machines handle what reviewers used to catch), escape rate stable or declining. The gap is never permanently closed — it is managed.

---

7. ADAPTATION

Feedback loop: measurement feeds back to intent. If rework is climbing, the thinking was shallow. If escapes are rising, the gates are underbuilt. If cost is climbing, check which vertex is failing first.

Capability Ladder progression:

Level	What the organization has built
0 – Unaware	No AI tools
1 – Experimenting	Individual exploration, no policy

Level	What the organization has built
2 – Assisted	Bounded tasks, individual judgment
3 – Integrated	AI in workflow, verification not scaled (most orgs are here)
4 – Governed	Gate tiers enforced, risk tiers defined, review budgeted
5 – Calibrated	Human judgment at two points, LLM-as-judge, overnight agents viable
6 – Adaptive	Agile at AI speed – triads, daily iteration, throwaway prototyping
7 – Agentic Operations	Agents own workflows end-to-end, supervised by monitoring and judges

The principle: this is agile, updated for AI. Same principles — cross-functional teams, tight iteration, working software over comprehensive documentation. AI changed the clock speed. The infrastructure that makes fast iteration trustworthy is the difference between speed and chaos.