AI SDLC — Project Genesis: Feature Vector Decomposition

Version: 1.8.0 **Date**: 2026-02-22 **Derived From**: AISDLC_IMPLEMENTATION_REQUIREMENTS.md (v3.12.0) **Method**: Asset Graph Model §6.4 (Task Planning as Trajectory Optimisation)

Purpose

Decompose INT-AISDLC-001 (AI SDLC Methodology Implementation) into feature vectors that trace trajectories through the asset graph. This is the prerequisite for design — per the methodology, features are identified before architecture is drawn.

Feature Vectors

REQ-F-ENGINE-001: Asset Graph Engine

The core graph topology, iteration function, and convergence/promotion mechanism.

Satisfies: REQ-GRAPH-001, REQ-GRAPH-002, REQ-GRAPH-003, REQ-ITER-001, REQ-ITER-002, REQ-ITER-003

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - Asset type registry with typed interfaces and Markov criteria - Admissible transition registry (directed, cyclic, extensible) - `iterate(Asset<Tn>, Context[], Evaluators) → Asset<Tn.k+1>` - `stable()` convergence check with configurable ε per evaluator - Promotion: candidate → Markov object when all evaluators pass

Dependencies: None — this is the foundation.

REQ-F-EVAL-001: Evaluator Framework

The three evaluator types and their composition per edge.

Satisfies: REQ-EVAL-001, REQ-EVAL-002, REQ-EVAL-003

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - Human evaluator interface (judgment, approval/rejection) - Agent(intent, context) evaluator interface (LLM-based delta computation) - Deterministic Test evaluator interface (pass/fail) - Evaluator composition registry: edge type → set of evaluators - Human accountability: AI assists, human decides

Dependencies: REQ-F-ENGINE-001.|design⟩ (evaluators plug into the iteration engine)

REQ-F-CTX-001: Context Management

Context[] as constraint surface, hierarchy, and spec reproducibility.

Satisfies: REQ-CTX-001, REQ-CTX-002, REQ-INTENT-004

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - Context store: ADRs, data models, templates, policy, graph topology, prior implementations - Hierarchical composition: global → org → team → project (later overrides earlier) - Context version control - Spec canonical serialisation (deterministic, content-addressable hash) - Spec immutability: evolution produces new versions, not mutations

Dependencies: REQ-F-ENGINE-001.|design⟩ (context feeds into iterate())

REQ-F-TRACE-001: Feature Vector Traceability

Intent capture, REQ keys, trajectories, dependencies, and task planning.

Satisfies: REQ-INTENT-001, REQ-INTENT-002, REQ-FEAT-001, REQ-FEAT-002, REQ-FEAT-003

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - Intent capture (INT-* format, structured, persisted) - Intent + Context[] → Spec composition - REQ key format and propagation across all graph assets - Bidirectional navigation (intent → runtime, runtime → intent) - Cross-feature dependency tracking - Task graph generation from feature decomposition + dependency compression

Dependencies: REQ-F-ENGINE-001.|code⟩ (needs graph engine), REQ-F-CTX-001.|design⟩ (needs context model)

REQ-F-EDGE-001: Edge Parameterisations

TDD, BDD, ADR, and code tagging configurations for common graph edges.

Satisfies: REQ-EDGE-001, REQ-EDGE-002, REQ-EDGE-003, REQ-EDGE-004

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - TDD co-evolution pattern (RED/GREEN/REFACTOR/COMMIT) at Code ↔ Tests edges - BDD Given/When/Then at Design → Test Cases and Design → UAT Tests edges - ADR generation at Requirements → Design edge - Code tagging: `Implements: REQ-*` / `Validates: REQ-*` (platform-agnostic tag format) - All parameterisations are evaluator configurations, not separate engines

Dependencies: REQ-F-EVAL-001.lcode⟩ (edge params configure evaluators)

REQ-F-LIFE-001: Full Lifecycle Closure

CI/CD, telemetry, homeostasis, feedback loop, and eco-intent generation.

Satisfies: REQ-LIFE-001, REQ-LIFE-002, REQ-LIFE-003, REQ-LIFE-004, REQ-LIFE-005, REQ-LIFE-006, REQ-LIFE-007, REQ-LIFE-008, REQ-LIFE-009, REQ-LIFE-010, REQ-LIFE-011, REQ-LIFE-012, REQ-INTENT-003

Trajectory: lreq⟩ → ldesign⟩ → lcode⟩ ↔ ltests⟩ → luat⟩

What converges: - CI/CD as graph edge (Code → CI/CD → Running System) - Telemetry tagged with REQ keys (`req="REQ-*"` as structured field) - Per-feature observability: latency, error rate, incidents queryable by REQ key - Homeostasis: is running system within constraint bounds? - Deviation → new INT-* intent → back into the graph - Eco-intent: automatic intent generation from ecosystem changes - Intent events as first-class objects (`intent_raised` with causal chain) - Signal source classification (7 types: gap, test_failure, refactoring, source_finding, process_gap, runtime_feedback, ecosystem) - Spec change events (`spec_modified` with trigger traceability, feedback loop detection) - Protocol enforcement hooks (mandatory side effects verified at every iteration boundary — reflex phase) - Spec review as gradient check (`delta(workspace, spec) → intents` — stateless, idempotent, affect-triaged) - Dev observer agent: markdown agent spec triggered by hooks, watches events.jsonl, computes delta(workspace, spec) → intents - CI/CD observer agent: markdown agent spec triggered after pipeline completion, maps build failures to REQ keys - Ops observer agent: markdown agent spec on schedule/alert, reads production telemetry, correlates with REQ keys - All observers are Markov objects: read inputs, emit events, no shared mutable state (actor model — event log is mailbox)

Dependencies: REQ-F-ENGINE-001.lcode⟩, REQ-F-TRACE-001.lcode⟩ (needs graph + REQ key propagation)

REQ-F-SENSE-001: Sensory Systems

Continuous interoceptive and exteroceptive monitoring with affect triage pipeline, running as a long-running service with review boundary for draft-only autonomy.

Satisfies: REQ-SENSE-001, REQ-SENSE-002, REQ-SENSE-003, REQ-SENSE-004, REQ-SENSE-005, REQ-SENSE-006

Trajectory: lreq⟩ → ldesign⟩ → lcode⟩ ↔ ltests⟩

What converges: - Sensory service architecture: long-running service with workspace watcher, monitor scheduler, affect triage - Interoceptive monitor framework (INTRO-001..007): event freshness, vector stall detection, test health, STATUS freshness, build health, spec/code drift, event log integrity - Exteroceptive monitor framework (EXTRO-001..004): dependency freshness, CVE scanning, runtime telemetry deviation, API contract changes - Affect triage pipeline: rule-based + agent-classified (tiered), severity weighting, escalation decision, profile-tunable thresholds - Homeostatic responses: probabilistic agent generates draft proposals only (no file modifications) - Review boundary: tool interface separates autonomous sensing from human-approved changes - New event types: `interoceptive_signal`, `exteroceptive_signal`, `affect_triage`, `draft_proposal` in events.jsonl - Monitor registry: configurable per project and per projection profile (`sensory_monitors.yml`, `affect_triage.yml`) - Monitor health: meta-monitoring (senses that sensing has failed) - Monitor/telemetry separation: Genesis produces events (sensing), genesis_monitor consumes telemetry (observing)

Dependencies: REQ-F-LIFE-001.lcode⟩ (needs gradient mechanics and event sourcing), REQ-F-EVAL-001.lcode⟩ (affect triage uses evaluator pattern)

REQ-F-TOOL-001: Developer Tooling

Plugin architecture, workspace, commands, release, test gap analysis, hooks, scaffolding, snapshots.

Satisfies: REQ-TOOL-001, REQ-TOOL-002, REQ-TOOL-003, REQ-TOOL-004, REQ-TOOL-005, REQ-TOOL-006, REQ-TOOL-007, REQ-TOOL-008, REQ-TOOL-009, REQ-TOOL-010, REQ-TOOL-011, REQ-TOOL-012, REQ-TOOL-013, REQ-TOOL-014

Trajectory: lreq⟩ → ldesign⟩ → lcode⟩ ↔ ltests⟩

What converges: - Plugin architecture: installable, discoverable, versioned methodology delivery - Developer workspace: task tracking, context preservation, git-integrated - Workflow commands: task CRUD, checkpoint/restore, status/coverage - Release management: semver, changelog, REQ key coverage in release notes - Test gap analysis: REQ keys vs tests, uncovered trajectories - Methodology hooks: commit/transition/session triggers, REQ tag validation - Project scaffolding: graph config, context dirs, workspace templates - Context snapshot: immutable session capture for recovery - Feature views: per-REQ-key cross-artifact status (grep-based traceability) - Spec/Design boundary enforcement: technology leakage detection, multiple design variants

Dependencies: REQ-F-ENGINE-001.ldesign⟩ (tooling wraps the engine), REQ-F-TRACE-001.ldesign⟩ (tooling uses REQ keys)

REQ-F-UX-001: User Experience

Two-command UX layer: state-driven routing (Start), project-wide observability (Status), progressive disclosure, automatic feature/edge selection, recovery and self-healing, human gate awareness (escalation notification), edge zoom management.

Satisfies: REQ-UX-001, REQ-UX-002, REQ-UX-003, REQ-UX-004, REQ-UX-005, REQ-UX-006, REQ-UX-007

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - State-driven routing: 8-state machine derived from workspace filesystem + event log - Progressive disclosure: ≤5 inputs at init, constraints deferred to design edge - Project-wide observability: "you are here" indicators, cross-feature rollup, signals, health - Automatic feature/edge selection: priority-based selection, topological edge walk - Recovery and self-healing: detect and guide recovery from inconsistent workspace states - Human gate awareness: escalation notification channels, pending review queue, zero-query awareness via status - Edge zoom management: expand edges to sub-graphs, collapse sub-graphs, selective zoom with mandatory waypoints

Dependencies: REQ-F-TOOL-001.|design⟩ (UX layer wraps tooling commands), REQ-F-ENGINE-001.|design⟩ (state detection reads graph)

REQ-F-COORD-001: Multi-Agent Coordination

Event-sourced agent coordination: agent identity, feature assignment via claims, work isolation with promotion gates, Markov-aligned parallelism, role-based evaluator authority.

Satisfies: REQ-COORD-001, REQ-COORD-002, REQ-COORD-003, REQ-COORD-004, REQ-COORD-005

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - Agent identity: `agent_id` + `agent_role` on all events, self-declared via registry - Feature assignment via events: `edge_claim` → serialiser → `edge_started` or `claim_rejected`; no lock files - Inbox/serialiser: agents emit to private inbox, single-writer serialiser resolves claims and appends to events.jsonl - Work isolation: agent drafts in `agents/<id>/drafts/`, promotion via evaluator gate + human review - Markov-aligned parallelism: inner product determines safe parallel assignment - Role-based authority: roles define convergence scope, escalation for out-of-scope edges

Dependencies: REQ-F-ENGINE-001.|code⟩ (event sourcing infrastructure), REQ-F-EVAL-001.|code⟩ (evaluator framework for authority scoping), REQ-F-TOOL-001.|code⟩ (start/status commands need multi-agent awareness)

REQ-F-SUPV-001: IntentEngine Formalization

The universal observer/evaluator composition law — fractal processing on every edge, ambiguity classification, three output types, chaining with affect propagation, consciousness-as-relative.

Satisfies: REQ-SUPV-001, REQ-SUPV-002 | REQ-SUPV-003 | REQ-F-SENSE-001 | | |

Trajectory: |req⟩ → |design⟩ → |code⟩ ↔ |tests⟩

What converges: - IntentEngine interface: `observer → evaluator → typed_output(reflex.log | specEventLog | escalate)` parameterised by intent+affect - Ambiguity classification: zero (reflex), bounded nonzero (probabilistic), persistent (escalate) — maps to three evaluator types - Three output types exhaustively classify all possible observation outcomes — mapped to existing event types - Observer/evaluator on every edge: every edge traversal is an IntentEngine invocation, producing classified observation + typed output - Chaining: one unit's output becomes next unit's intent+affect — the supervision hierarchy - Affect propagation: urgency/valence carried forward and transformed at each level - Consciousness-as-relative: Level N's escalate = Level N+1's reflex; Level N's reflex.log invisible to Level N+1 - Fractal application table: single iteration → edge → feature → sensory → production → spec review - Constraint tolerances: every constraint has a measurable threshold; tolerances make the gradient operational; breach → optimization intent

Dependencies: REQ-F-ENGINE-001.ldesign⟩ (IntentEngine composes over the four primitives), REQ-F-EVAL-001.ldesign⟩ (ambiguity classification maps to evaluator types)

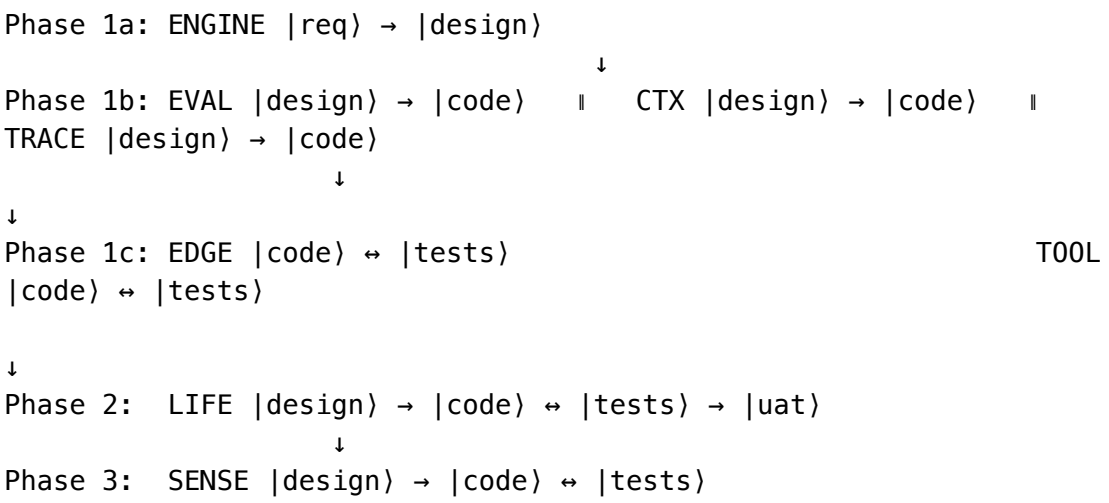
Dependency Graph

```
REQ-F-ENGINE-001 (Asset Graph Engine)
    │
    ├──→ REQ-F-EVAL-001 (Evaluator Framework)
    │         │
    │         ├──→ REQ-F-EDGE-001 (Edge Parameterisations)
    │         │
    │         ├──→ REQ-F-SUPV-001 (IntentEngine Formalization) ←── also
depends on ENGINE
    │         │
    │         └──→ REQ-F-SENSE-001 (Sensory Systems) ←── also depends
on LIFE-001
    │
    ├──→ REQ-F-CTX-001 (Context Management)
    │
    ├──→ REQ-F-TRACE-001 (Feature Traceability)
    │         │
    │         └──→ REQ-F-LIFE-001 (Lifecycle Closure)
    │                   │
    │                   └──→ REQ-F-SENSE-001 (Sensory Systems)
    │
    └──→ REQ-F-TOOL-001 (Developer Tooling)
              │
              ├──→ REQ-F-UX-001 (User Experience)
              │
              └──→ REQ-F-COORD-001 (Multi-Agent Coordination) ←── also
depends on ENGINE, EVAL
```

Parallel work (zero inner product — independent once ENGINE.ldesign⟩ converges): - REQ-F-EVAL-001 ∥ REQ-F-CTX-001 ∥ REQ-F-TRACE-001

Sequential constraints: - ENGINE.ldesign⟩ < EVAL.lcode⟩ (evaluators need engine interface) - ENGINE.ldesign⟩ < CTX.lcode⟩ (context needs engine interface) - EVAL.lcode⟩ < EDGE.lcode⟩ (edge params configure evaluators) - TRACE.lcode⟩ < LIFE.lcode⟩ (lifecycle needs REQ key propagation) - LIFE.lcode⟩ + EVAL.lcode⟩ < SENSE.lcode⟩ (sensory systems need gradient mechanics + evaluator pattern) - ENGINE.ldesign⟩ + TRACE.ldesign⟩ < TOOL.lcode⟩ (tooling wraps both)

Task Graph (Compressed)



ENGINE design is the critical path. Once it converges, three features parallelise.

Coverage Check

Implementation Requirement	Feature Vector
REQ-INTENT-001	REQ-F-TRACE-001
REQ-INTENT-002	REQ-F-TRACE-001
REQ-INTENT-003	REQ-F-LIFE-001
REQ-INTENT-004	REQ-F-CTX-001
REQ-GRAPH-001	REQ-F-ENGINE-001
REQ-GRAPH-002	REQ-F-ENGINE-001
REQ-GRAPH-003	REQ-F-ENGINE-001
REQ-ITER-001	REQ-F-ENGINE-001
REQ-ITER-002	REQ-F-ENGINE-001
REQ-ITER-003	REQ-F-ENGINE-001

Implementation Requirement	Feature Vector
REQ-EVAL-001	REQ-F-EVAL-001
REQ-EVAL-002	REQ-F-EVAL-001
REQ-EVAL-003	REQ-F-EVAL-001
REQ-CTX-001	REQ-F-CTX-001
REQ-CTX-002	REQ-F-CTX-001
REQ-FEAT-001	REQ-F-TRACE-001
REQ-FEAT-002	REQ-F-TRACE-001
REQ-FEAT-003	REQ-F-TRACE-001
REQ-LIFE-001	REQ-F-LIFE-001
REQ-LIFE-002	REQ-F-LIFE-001
REQ-LIFE-003	REQ-F-LIFE-001
REQ-EDGE-001	REQ-F-EDGE-001
REQ-EDGE-002	REQ-F-EDGE-001
REQ-EDGE-003	REQ-F-EDGE-001
REQ-EDGE-004	REQ-F-EDGE-001
REQ-TOOL-001	REQ-F-TOOL-001
REQ-TOOL-002	REQ-F-TOOL-001
REQ-TOOL-003	REQ-F-TOOL-001
REQ-TOOL-004	REQ-F-TOOL-001
REQ-TOOL-005	REQ-F-TOOL-001
REQ-TOOL-006	REQ-F-TOOL-001
REQ-TOOL-007	REQ-F-TOOL-001
REQ-TOOL-008	REQ-F-TOOL-001
REQ-TOOL-009	REQ-F-TOOL-001
REQ-TOOL-010	REQ-F-TOOL-001
REQ-TOOL-011	REQ-F-TOOL-001
REQ-TOOL-012	REQ-F-TOOL-001

Implementation Requirement	Feature Vector
REQ-TOOL-013	REQ-F-TOOL-001
REQ-TOOL-014	REQ-F-TOOL-001
REQ-LIFE-004	REQ-F-LIFE-001
REQ-LIFE-005	REQ-F-LIFE-001
REQ-LIFE-006	REQ-F-LIFE-001
REQ-LIFE-007	REQ-F-LIFE-001
REQ-LIFE-008	REQ-F-LIFE-001
REQ-LIFE-009	REQ-F-LIFE-001
REQ-LIFE-010	REQ-F-LIFE-001
REQ-LIFE-011	REQ-F-LIFE-001
REQ-LIFE-012	REQ-F-LIFE-001
REQ-SENSE-001	REQ-F-SENSE-001
REQ-SENSE-002	REQ-F-SENSE-001
REQ-SENSE-003	REQ-F-SENSE-001
REQ-SENSE-004	REQ-F-SENSE-001
REQ-SENSE-005	REQ-F-SENSE-001
REQ-SENSE-006	REQ-F-SENSE-001
REQ-UX-001	REQ-F-UX-001
REQ-UX-002	REQ-F-UX-001
REQ-UX-003	REQ-F-UX-001
REQ-UX-004	REQ-F-UX-001
REQ-UX-005	REQ-F-UX-001
REQ-UX-006	REQ-F-UX-001
REQ-UX-007	REQ-F-UX-001
REQ-COORD-001	REQ-F-COORD-001
REQ-COORD-002	REQ-F-COORD-001
REQ-COORD-003	REQ-F-COORD-001

Implementation Requirement	Feature Vector
REQ-COORD-004	REQ-F-COORD-001
REQ-COORD-005	REQ-F-COORD-001
REQ-SUPV-001	REQ-F-SUPV-001
REQ-SUPV-002	REQ-F-SUPV-001
REQ-SUPV-003	REQ-F-SENSE-001

69/69 requirements covered. No orphans.

Summary

Feature Vector	Impl Reqs	Phase	Dependencies
REQ-F-ENGINE-001	6	1a	None
REQ-F-EVAL-001	3	1b	ENGINE
REQ-F-CTX-001	3	1b	ENGINE
REQ-F-TRACE-001	5	1b	ENGINE, CTX
REQ-F-EDGE-001	4	1c	EVAL
REQ-F-LIFE-001	13	2	ENGINE, TRACE
REQ-F-SENSE-001	7	3	LIFE, EVAL
REQ-F-TOOL-001	10	1c	ENGINE, TRACE
REQ-F-UX-001	7	1c	TOOL, ENGINE
REQ-F-COORD-001	5	2	ENGINE, EVAL, TOOL
REQ-F-SUPV-001	2	1b	ENGINE, EVAL
Total	**69**		

11 feature vectors. 69 implementation requirements. Full coverage. Critical path: ENGINE design.