

AI SDLC — Feature Vector Decomposition

Version: 1.1.0 **Date:** 2026-02-20 **Derived From:**

[AISDLC IMPLEMENTATION REQUIREMENTS.md](#) (v3.2.0) **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-GRAFH-001, REQ-GRAFH-002, REQ-GRAFH-003, REQ-ITER-001, REQ-ITER-002

Trajectory: $\lvert \text{req} \rangle \rightarrow \lvert \text{design} \rangle \rightarrow \lvert \text{code} \rangle \leftrightarrow \lvert \text{tests} \rangle$

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: $\lvert \text{req} \rangle \rightarrow \lvert \text{design} \rangle \rightarrow \lvert \text{code} \rangle \leftrightarrow \lvert \text{tests} \rangle$

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.`ldesign` (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: `lreq` → `ldesign` → `lcode` ↔ `ltests`

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.`ldesign` (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: `lreq` → `ldesign` → `lcode` ↔ `ltests`

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.`lcode` (needs graph engine), REQ-F-CTX-001.`ldesign` (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: `lreq` → `ldesign` → `lcode` ↔ `ltests`

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-INTENT-003

Trajectory: |req> → |design> → |code> ↔ |tests> → |uat>

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

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

REQ-F-TOOL-001: Developer Tooling

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

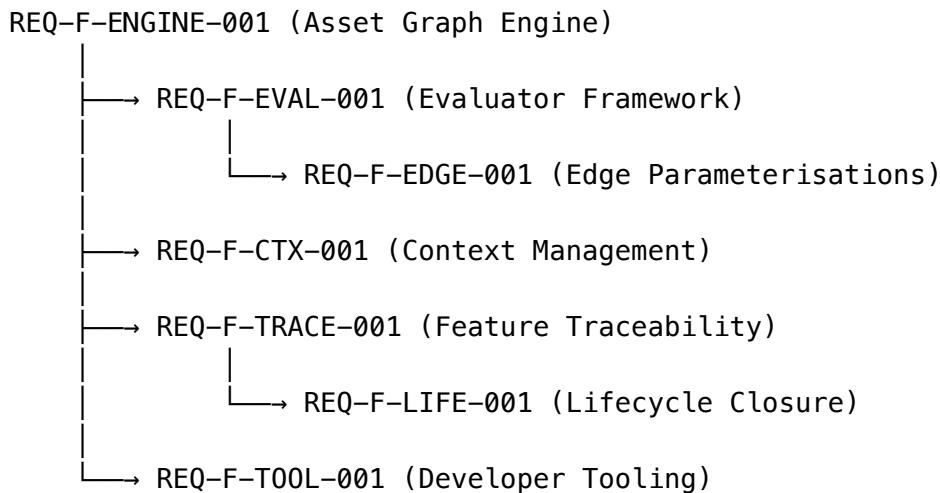
Satisfies: REQ-TOOL-001, REQ-TOOL-002, REQ-TOOL-003, REQ-TOOL-004, REQ-TOOL-005, REQ-TOOL-006, REQ-TOOL-007, REQ-TOOL-008, REQ-TOOL-009, REQ-TOOL-010

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

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)

Dependency Graph



Parallel work (zero inner product — independent once ENGINE.l.design 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) - `ENGINE.ldesign` + `TRACE.ldesign` < `TOOL.lcode` (tooling wraps both)

Task Graph (Compressed)

Phase 1a: ENGINE $|req\rangle \rightarrow |design\rangle$

Phase 1b: EVAL |design) → |code) || CTX |design) → |code) ||
TRACE |design) → |code)

Phase 1c: EDGE $|code\rangle \leftrightarrow |tests\rangle$
 $|code\rangle \leftrightarrow |tests\rangle$

TOOL

↓
Phase 2: LIFE |design) → |code) ↔ |tests) → |uat)

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 |

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

| Implementation Requirement | Feature Vector |
|-----------------------------------|-----------------------|
| 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-LIFE-004 | REQ-F-LIFE-001 |

35/35 requirements covered. No orphans.

Summary

| Feature Vector | Impl Reqs | Phase | Dependencies |
|-----------------------|------------------|--------------|---------------------|
| REQ-F-ENGINE-001 | 5 | 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 | 5 | 2 | ENGINE, TRACE |
| REQ-F-TOOL-001 | 10 | 1c | ENGINE, TRACE |
| Total | 35 | | |

7 feature vectors. 35 implementation requirements. Full coverage. Critical path: ENGINE design.