AI SDLC — Executive Summary

A condensed overview of the Asset Graph Model — the formal system underlying all platform-specific implementations.

1. The Formal System: 4 Primitives, 1 Operation

The methodology is defined by four invariants that must hold in every valid projection:

Primitive	Definition
Graph	A topology of typed assets connected by admissible transitions.
Iterate	The only operation: `iterate(asset, context, evaluators) → candidate`.
Evaluators	Convergence criteria ({Human, Agent, Tests}) that compute δ.
Spec + Context	The constraint surface that bounds construction.

The Operation

Every edge in the graph is traversed by calling `iterate()` until evaluators report convergence ($\delta < \epsilon$).

2. The Asset Graph: Features as Trajectories

The SDLC is not a pipeline; it is a **Directed Cyclic Graph**. * **Composite Vectors**: A feature is a trajectory (a vector) through the graph. It is identified by a **REQ key** that threads through every asset it produces. * **Markov Objects**: Converged assets are conditionally independent. You interact with them through their **Boundary** (interfaces/contracts), not their construction history. * **Zoomable**: Any edge can be expanded into a sub-graph (e.g., Design → Code expanding into Module Decomposition) or collapsed into a single edge.

3. The IntentEngine: Homeostasis

The system is a self-regulating organism that closes the loop between construction and observation.

- **Computed Intent**: Intent is not an external command; it is a generated output of any observer detecting a delta: `intent = delta(observed_state, spec)`.
- **Three Processing Phases**:
  1. **Reflex (Autonomic)**: Event emission, test execution, protocol hooks. Fires unconditionally.
  2. **Affect (Limbic)**: Signal classification and triage. Assigns urgency and determines escalation.
  3. **Conscious (Deliberative)**: Human/Agent judgment, intent generation, spec modification.
- **Sensory Systems**:
  - **Interoception**: Continuous monitoring of internal health (e.g., test staleness, stalled vectors).
  - **Exteroception**: Continuous monitoring of the environment (e.g., CVEs, dependency updates).

4. Execution Model: Functor Escalation ($F_D \rightarrow F_P \rightarrow F_H$)

Every functional unit (Evaluate, Construct, Sense) has three renderings that escalate based on ambiguity:

1. **Deterministic** (F_D): Zero ambiguity. Cheap, fast (e.g., compilers, linters).
2. **Probabilistic** (F_P): Bounded ambiguity. Moderate cost (e.g., LLM/Agent reasoning).
3. **Human** (F_H): Persistent ambiguity. High cost (e.g., human approval/judgment).

5. Projections: Selective Capability Enablement

A **Projection** is a valid instance of the formal system that selectively enables capabilities while preserving all four invariants. * **Profiles**: Named configurations (Minimal, Standard, Full) that define which edges exist, which evaluators are active, and what context is required. * **Logical Completeness**: The full system is always present in the specification; projections omit edges, evaluators, or context that a given project does not require.

6. Multi-Tenant by Design

The methodology is platform-agnostic. The specification defines WHAT; each implementation (`imp_<name>/`) binds HOW to a specific platform. One spec, many designs — each with its own architecture, tooling, and deployment model.

7. Protocol Enforcement

Observability is a constraint, not a feature. The system uses **Reflex Hooks** to verify that every iteration emits an event and updates the feature vector state. If the bookkeeping is skipped, the construction is blocked.