**ALAN IDE Development Roadmap**

**Success-Oriented Execution Plan: Phases 2–6**

Each phase below is broken into clear, sequential sprints, team tracks, and interface modules. Each word is chosen to ensure clarity, actionability, and success. This roadmap is your blueprint to deliver the world's first post-symbolic cognitive IDE.

**🚀 Phase 2: Interactive Cognitive IDE Prototype**

**Goal:** Transition ALAN from a kernel to a usable, field-driven development interface.

**✅ Sprints**

1. **Field Canvas Prototype (2 weeks)**
   * Create the semantic layout engine
   * Implement basic node placement in Euclidean/Hyperbolic space
   * Draw live phase-colored concept nodes + coupling lines
2. **Koopman Panel + Entropy Overlay (1 week)**
   * Show live eigenvalues, entropy plot, attractor label
   * Clickable Koopman mode selector
3. **Morph Console & Controls (1 week)**
   * Add real-time morphing by Koopman mode
   * Epsilon slider and morph replay buffer
4. **Agent Framework (2 weeks)**
   * Refactor Agent: detect clusters, suggest modularization
   * Debug Advisor: detect phase drift, spectral anomalies
   * Doc Agent: create and update conceptual summaries
5. **Concept Capsule Editor (2 weeks)**
   * Node editor with code, phase history, morph controls
   * Coupling map and local resonance graph
6. **ScholarSphere Memory Alpha (1 week)**
   * Store and replay morph histories
   * Detect prior morph analogies

**💼 Owner Tracks**

* **Field Dynamics:** Resonance, phase, Koopman, morphing
* **UI/UX Canvas:** Layout, interaction, annotations
* **Agent Core:** Agent triggers, action models, dashboards
* **Memory:** Morph logging, analogical pattern mining

**🔹 Interface Modules**

* ConceptFieldCanvas
* KoopmanSpectrumPanel
* MorphControlDock
* CognitiveAgentDock
* ConceptCapsuleEditor
* ScholarSphereMemory

**📊 Phase 3: Developer-Facing MVP (Alpha)**

**Goal:** Create a fully usable standalone IDE for real-world semantic coding.

**✅ Sprints**

1. **Code Import & Concept Extraction (2 weeks)**
   * Parse real Python projects into ALAN's concept graph
   * Create stable IDs and inheritance-aware links
2. **Live Code Editing & Field Sync (2 weeks)**
   * Bi-directional updates between code and concept graph
   * Track resonance impact of local edits
3. **Execution Monitoring (2 weeks)**
   * Link code execution traces to phase updates
   * Detect real-time divergence and instability
4. **Refactor Tools (2 weeks)**
   * Agent-led code movement and structure adaptation
   * Morph-driven suggestions to consolidate or isolate modules
5. **Export Tools (1 week)**
   * Convert field state to updated source code
   * Track lineage of semantic edits

**💼 Owner Tracks**

* **Parser & Language Adapters**
* **Bidirectional Sync**
* **Execution Field Tracer**
* **Agent Suggestion UX**
* **Exporter**

**🔹 Interface Modules**

* LiveCodeEditor
* PhaseLinkedExecutionGraph
* FieldRefactorPanel
* ExportComposer

**🔧 Phase 4: Large Project & Multi-Language Expansion**

**Goal:** Support industrial-scale codebases with seamless resonance.

**✅ Sprints**

1. **Language Adapter: TypeScript & Rust (3 weeks)**
   * Shared semantic primitives mapped into ALAN space
   * Convert call graphs and types into concept relationships
2. **Concept Graph Partitioning (2 weeks)**
   * Hierarchical resonance layers
   * Load-on-focus and spectral indexing
3. **CI/CD Integration (2 weeks)**
   * Run agents on pull requests
   * Report morph deltas, resonance shifts, and field health
4. **Multi-Language Morph Tracking (2 weeks)**
   * Watch morphs cascade across Python → Rust → JS
   * Suggest cross-language refactors based on resonance

**💼 Owner Tracks**

* **Language Adapter Team**
* **Field Scaling / Memory Optimization**
* **CI Agent Integration**
* **CrossLang Morph Coordinator**

**🔹 Interface Modules**

* LanguageAdapterLoader
* ResonanceLayerController
* CIIntegrationDashboard
* CrossLangMorphPanel

**🔄 Phase 5: Adaptive IDE (Beta → 1.0)**

**Goal:** Enable ALAN to evolve its field model alongside the developer.

**✅ Sprints**

1. **Spectral Pattern Learner (3 weeks)**
   * Learn user-preferred morph modes
   * Detect repeating attractor states across sessions
2. **Agent Feedback Loops (2 weeks)**
   * Agents learn from user overrides
   * Adjust morph thresholds and intervention sensitivity
3. **Field Stability Profiler (2 weeks)**
   * Visualize long-term entropy + stability trendlines
   * Predict phase instability before it occurs
4. **User Rhythm Modeling (2 weeks)**
   * Build developer-specific resonance fingerprints
   * Predict likely concept drift and morph needs

**💼 Owner Tracks**

* **Pattern Learner Team**
* **Field Profiler Devs**
* **Adaptive Agent Core**

**🔹 Interface Modules**

* SpectralHabitModel
* FieldPredictorHUD
* AgentLearningConsole

**🌌 Phase 6: AI-Collaborative & Research Platform**

**Goal:** Establish ALAN as the backbone for semantic collaboration, intelligent synthesis, and cognitive interface R&D.

**✅ Sprints**

1. **Multi-User Field Collaboration (3 weeks)**
   * Share concept graphs with phase-separated user overlays
   * Merge/morph across collaborators
2. **LLM-Guided Concepts (2 weeks)**
   * Inject LLM-generated suggestions into Koopman-mode alignments
   * Validate through field stability and phase retention
3. **Language-Field Translations (2 weeks)**
   * Describe morphs and phase shifts in natural language
   * Generate commit summaries from attractor transitions
4. **Research Mode Toolkit (2 weeks)**
   * Export resonance data for ML research
   * Hook into neuroscience/HCI experiments

**💼 Owner Tracks**

* **Collaboration & Sync Engine**
* **LLM Interface Team**
* **Field/Natural Language Bridge**
* **Research Data Export & Instrumentation**

**🔹 Interface Modules**

* FieldCollaborationOverlay
* LLMSemanticBridge
* AttractorSummaryFeed
* FieldResearchExporter

Each phase and sprint above is structured to ensure consistent delivery, testable checkpoints, and meaningful developer/user cognition at every step. ALAN will not only meet expectations—it will redefine what development feels like.