Contents

Cognitive Architecture

Beyond Orchestration: Structuring Thought

RADIANT's Cognitive Architecture moves beyond simple model orchestration into true cognitive structuring. These five capabilities enable results that are categorically superior to single-model approaches.

Overview

Feature	Purpose	Key Benefit
Tree of Thoughts	System 2 reasoning	Solves problems single-shot models can't
GraphRAG	Knowledge mapping	Multi-hop reasoning across documents
Deep Research	Background agents	50+ source analysis in 30 minutes
Dynamic LoRA	Hot-swap expertise	Specialist-level domain performance
Generative UI	App factory	AI becomes the interface

---

1. Tree of Thoughts (System 2 Reasoning)

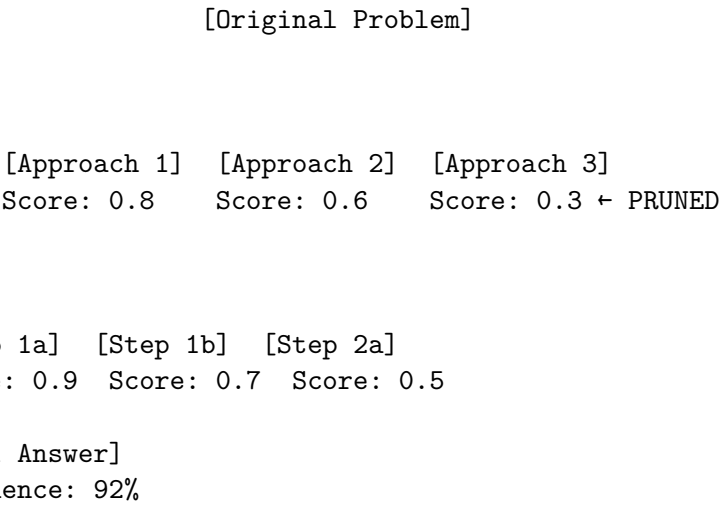
The Problem

Standard LLMs operate on "System 1" thinking—fast, intuitive, linear. They write the first word that comes to mind. If they make a logic error in step 1, the entire chain collapses.

The Solution

Implement Monte Carlo Tree Search (MCTS) or Beam Search for deliberate reasoning.

How It Works



1. **Branch**: Generate 3 distinct "first steps" for a complex problem
2. **Evaluate**: Use a scoring model to rate which path is most promising

3. **Backtrack**: If a path scores poorly, rewind and try a different branch
4. **Converge**: Best path becomes the final answer

Configuration

Setting	Default	Description
`maxDepth`	5	Maximum reasoning steps
`branchingFactor`	3	Thoughts per branch
`pruneThreshold`	0.3	Score below which to prune
`selectionStrategy`	beam	beam, mcts, or greedy
`beamWidth`	2	Top K paths to keep
`defaultThinkingTimeMs`	30000	Default thinking budget

User Experience

Users can "trade time for intelligence": - Quick answer: 10 seconds - Normal thinking: 30 seconds - Deep reasoning: 2 minutes - Extended analysis: 5 minutes

Best For

- Math problems
- Logic puzzles
- Multi-step planning
- Architecture decisions
- Code debugging

Key Files

- Types: `packages/shared/src/types/cognitive-architecture.types.ts`
- Service: `packages/infrastructure/lambda/shared/services/tree-of-thoughts.service.ts`
- Table: `reasoning_trees`

---

2. GraphRAG (Structured Knowledge Mapping)

The Problem

Standard RAG uses vector similarity. If you search "Apple," it finds text mathematically close to "Apple." It fails at multi-hop reasoning:

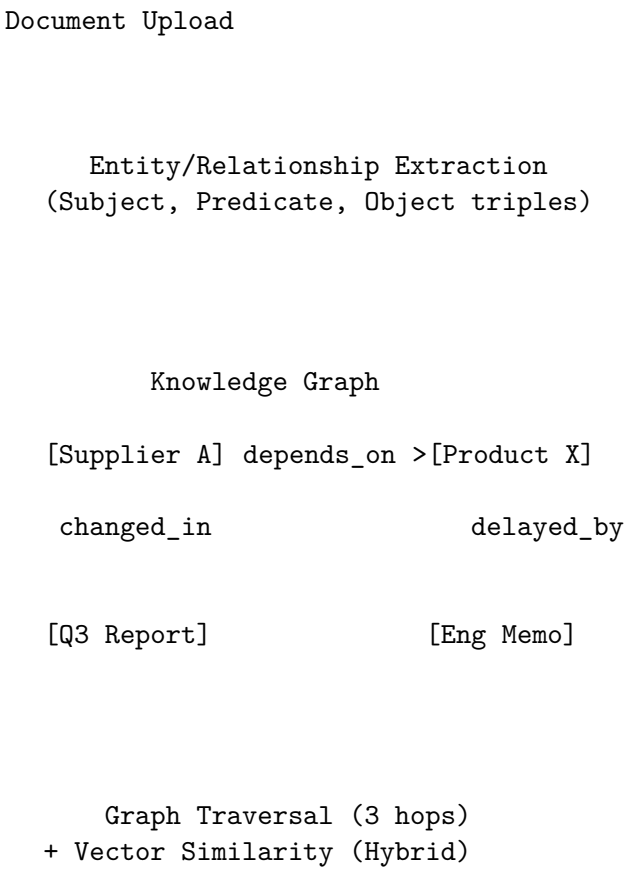
> "How does the supplier change in the Q3 report affect the delayed launch mentioned in the Engineering memo?"

Vector search can't connect these dots.

The Solution

Extract entities and relationships into a knowledge graph, then traverse connections.

How It Works



Entity Types

- `person`, `organization`, `document`, `concept`
- `event`, `location`, `product`, `technology`
- `metric`, `date`, `custom`

Relationship Types

- `authored_by`, `depends_on`, `blocked_by`, `related_to`
- `part_of`, `caused_by`, `precedes`, `follows`
- `mentions`, `contradicts`, `supports`, `defines`

Configuration

Setting	Default	Description
`maxEntitiesPerDocument`	50	Extraction limit
`maxRelationshipsPerDocument`	100	Relationship limit

Setting	Default	Description
`minConfidenceThreshold`	0.7	Quality filter
`enableHybridSearch`	true	Combine graph + vector
`graphWeight`	0.6	Weight for graph results
`vectorWeight`	0.4	Weight for vector results
`maxHops`	3	Traversal depth

Key Files

- Service: `packages/infrastructure/lambda/shared/services/graph-rag.service.ts`
- Tables: `knowledge_entities`, `knowledge_relationships`

---

3. Deep Research Agents

The Problem

Chat interfaces train users to expect answers in <10 seconds. This forces models to be shallow. Humans don't solve complex engineering problems in 10 seconds.

The Solution

Decouple "Request" from "Response" with fire-and-forget background research.

How It Works

```
User: "Map the competitive landscape of solid-state batteries"



        Job Queued
        ETA: 25 min



        Research Agent

    Phase 1: Planning (5%)
    - Generate search queries
    - Identify credible sources

    Phase 2: Gathering (10-50%)
    - Visit 50+ websites
    - Download PDFs
    - Follow relevant links

    Phase 3: Analyzing (50-80%)
    - Extract key information
```

```
- Score relevance
- Check credibility

Phase 4: Synthesizing (80-95%)
- Generate briefing document
- Extract key findings
- Formulate recommendations

Phase 5: Review (95-100%)
- Quality check
- Format output
```

```
Notification
"Research
 Complete"
```

```
   Briefing Document

# Competitive Landscape
## Executive Summary
## Key Findings (12)
## Recommendations (5)
## Sources (47)
```

Research Types

- `competitive_analysis` - Market competitors
- `market_research` - Trends, sizing, forecasts
- `technical_review` - Specifications, architecture
- `literature_review` - Academic papers
- `fact_check` - Verification
- `general` - Open-ended research

Configuration

Setting	Default	Description
`maxSources`	50	Sources to process
`maxDepth`	2	Link following depth
`maxDurationMs`	1800000	30 minute timeout
`parallelRequests`	5	Concurrent fetches

Setting	Default	Description
`requireCredibleSources`	true	Quality filter
`minSourceCredibility`	0.6	Credibility threshold

Key Files

- Service: `packages/infrastructure/lambda/shared/services/deep-research.service.ts`
- Tables: `research_jobs`, `job_queue`

---

4. Dynamic LoRA Swapping

The Problem

Generalist models (like Gemini Ultra) are "Jacks of all trades." They lack deep, niche expertise in specific domains like "Cobol Migration" or "California Property Law."

The Solution

Hot-swap lightweight LoRA adapters (~100MB each) that transform a generalist into a specialist.

How It Works

```
          User Query
"What are the easements requirements
 for commercial property in San Diego?"




       Domain Detection
Field: Law
Domain: Real Estate Law
Subspecialty: California Property




       LoRA Registry (S3)


 california_property_law.safetensor
 Size: 98MB
 Base: Llama-3-70B
 Rank: 32, Alpha: 64
 Benchmark: 0.94
```

```
   SageMaker Multi-Model Endpoint

Base Model: Llama-3-70B
+ california_property_law LoRA

Load Time: 1.2s (cached: 0ms)



        Expert-Level Response
```

Available Domains

- `legal` - Law specializations
- `medical` - Healthcare, clinical
- `financial` - Finance, economics
- `scientific` - Research domains
- `coding` - Programming languages
- `creative_writing` - Fiction, poetry
- `translation` - Languages
- `customer_support` - Support patterns
- `technical_writing` - Documentation

Configuration

Setting	Default	Description
`enabled`	false	Requires SageMaker setup
`registryBucket`	radiant-lora-adapters	S3 bucket
`cacheSize`	5	Adapters in memory
`maxLoadTimeMs`	5000	Load timeout
`fallbackToBase`	true	Use base on failure
`autoSelectByDomain`	true	Auto-select adapter

Key Files

- Service: `packages/infrastructure/lambda/shared/services/dynamic-lora.service.ts`
- Table: `lora_adapters`

---

5. Generative UI (App Factory)

The Problem

No matter how smart the AI, if the output is just Markdown text, the utility is limited.

The Solution

The AI generates the interface itself—interactive components that users can manipulate.

How It Works

```
User: "Compare pricing of GPT-4, Claude 3, and Gemini"

Traditional AI Response:

 Here's a comparison:

 | Model    | Input   | Output  |
 |----------|---------|---------|
 | GPT-4    | $30/M   | $60/M   |
 | Claude 3 | $15/M   | $75/M   |
 | Gemini   | $7/M    | $21/M   |


Generative UI Response:

          Pricing Calculator

  Input Tokens:         50,000
  Output Tokens:         25,000


    GPT-4             $2.25
    Claude 3           $2.63
    Gemini            $0.88


    Gemini is 61% cheaper for this load
```

Component Types

- `chart` - Bar, line, pie charts
- `table` - Interactive, sortable tables
- `calculator` - Input sliders, computed outputs
- `comparison` - Side-by-side comparisons
- `timeline` - Chronological events
- `form` - Input forms
- `diagram` - Flow diagrams
- `map` - Geographic displays
- `kanban` - Task boards
- `calendar` - Date displays

Auto-Detection Triggers

The system automatically generates UI when it detects: - “compare” → Comparison component - “calculate” → Calculator component - “visualize”, “chart”, “graph” → Chart component - “table” → Table component - “timeline” → Timeline component

Configuration

Setting	Default	Description
`enabled`	true	Enable Generative UI
`maxComponentsPerResponse`	3	Component limit
`autoDetectOpportunities`	true	Auto-generate
`defaultTheme`	auto	light, dark, auto

Key Files

- Service: `packages/infrastructure/lambda/shared/services/generative-ui.service.ts`
- Table: `generated_ui`

AGI Brain Integration

All five cognitive features integrate with the AGI Brain Planner:

```
// In agi-brain-planner.service.ts
async generatePlan(prompt: string): Promise<BrainPlan> {
  // Detect if Tree of Thoughts should be used
  if (this.shouldUseTreeOfThoughts(prompt)) {
    plan.orchestrationMode = 'extended_thinking';
    plan.cognitiveFeatures.push('tree_of_thoughts');
  }

  // Check if GraphRAG has relevant knowledge
  const graphContext = await graphRAGService.hybridSearch(prompt);
  if (graphContext.entities.length > 0) {
    plan.contextSources.push('knowledge_graph');
  }

  // Dispatch deep research for complex queries
  if (this.isResearchQuery(prompt)) {
    plan.asyncResearch = true;
  }

  // Select domain-specific LoRA
  const adapter = await dynamicLoRAService.selectAdapterForDomain(domain);
  if (adapter) {
    plan.loraAdapter = adapter.id;
  }
```

```
  // Detect UI generation opportunities
  const uiOpportunity = await generativeUIService.detectUIOpportunity(prompt);
  if (uiOpportunity.shouldGenerate) {
    plan.generateUI = uiOpportunity.suggestedTypes;
  }
}
```

Database Tables

Table	Purpose
`reasoning_trees`	Tree of Thoughts sessions
`knowledge_entities`	GraphRAG entities
`knowledge_relationships`	GraphRAG relationships
`research_jobs`	Deep Research job tracking
`job_queue`	Async job queue
`lora_adapters`	LoRA adapter registry
`generated_ui`	Generated UI components
`cognitive_architecture_config`	Per-tenant configuration

Admin Dashboard

Location: Settings → Cognitive Architecture

The admin dashboard provides: - Enable/disable toggles for each feature - Configuration sliders and inputs - Explanatory information panels - Per-tenant customization

---

Related Documentation

- AGI Brain Plan System
- Domain Taxonomy
- Intelligence Aggregator