Jotty V2 Architecture

Jotty Team

Contents

Jotty V2 - Comprehensive Architecture Diagram

System Overview

```
flowchart TB
    subgraph USER_LAYER["  USER INTERFACE LAYER"]
        USER[("  User")]
        CLI["JottyCLI<br/>cli/app.py"]
        RENDERER["UIRenderer<br/>cli/ui/renderer.py"]
    end

    subgraph COMMAND_LAYER["  COMMAND LAYER"]
        CMD_REGISTRY["CommandRegistry"]
        RUN_CMD["RunCommand<br/>cli/commands/run.py"]
        EXPORT_CMD["ExportCommand<br/>cli/commands/export.py"]
        RESUME_CMD["ResumeCommand<br/>cli/commands/resume.py"]
        JUSTJOT_CMD["JustJotCommand<br/>cli/commands/justjot.py"]
    end

    subgraph ORCHESTRATION_LAYER["  ORCHESTRATION LAYER (V2)"]
        SWARM_MGR["SwarmManager<br/>core/orchestration/v2/swarm_manager.py"]
        AGENT_RUNNER["AgentRunner<br/>core/orchestration/v2/agent_runner.py"]
        LEAN_EXEC["LeanExecutor<br/>core/orchestration/v2/lean_executor.py"]
        SWARM_INSTALL["SwarmInstaller<br/>core/orchestration/v2/swarm_installer.py"]
        subgraph TEMPLATES["  Swarm Templates"]
            TEMPLATE_REG["TemplateRegistry"]
            SWARM_LEAN["SwarmLean"]
            BASE_TEMPLATE["BaseTemplate"]
        end
    end

    subgraph AGENT_LAYER["  AGENT LAYER"]
        AGENTIC_PLANNER["AgenticPlanner<br/>core/agents/agentic_planner.py"]
        AUTO_AGENT["AutoAgent<br/>core/agents/auto_agent.py"]
        INSPECTOR["InspectorAgent<br/>(Architect/Auditor)"]

        subgraph PLANNING["  Planning"]
            TASK_INFERENCE["Task Type Inference<br/>RESEARCH|COMPARISON|<br/>CREATION|ANALYS
            WORKFLOW_PLAN["Workflow Planning<br/>Steps + Dependencies"]
        end
    end

    subgraph MEMORY_LAYER["  MEMORY SYSTEM (Brain-Inspired)"]
        CORTEX["MemoryCortex<br/>core/memory/cortex.py"]

        subgraph MEMORY_LEVELS["  5-Level Hierarchy"]
            EPISODIC["EPISODIC<br/>(fast decay)"]
            SEMANTIC["SEMANTIC<br/>(slow decay)"]
            PROCEDURAL["PROCEDURAL<br/>(medium decay)"]
            META["META<br/>(no decay)"]
            CAUSAL["CAUSAL<br/>(no decay)"]
        end

        subgraph MEMORY_COMPONENTS["  Memory Components"]
            LEVEL_CLASSIFIER["MemoryLevelClassifier<br/>(LLM-based)"]
            RAG_RETRIEVER["LLMRAGRetriever"]
            DEDUP_ENGINE["DeduplicationEngine"]
            CAUSAL_EXTRACT["CausalExtractor"]
            BRAIN_STATE["BrainStateMachine<br/>AWAKE|SLEEP"]
            HIPPO_EXTRACT["HippocampalExtractor"]
            RIPPLE_CONSOL["SharpWaveRipple<br/>Consolidator"]
        end
    end

    subgraph LEARNING_LAYER["  LEARNING SYSTEM (RL)"]
        TD_LEARNER["TDLambdaLearner<br/>Temporal Difference"]

        subgraph RL_COMPONENTS["  RL Components"]
```

```
        TRAJECTORY_PRED["LLMTrajectoryPredictor"]
        DIVERGENCE_MEM["DivergenceMemory"]
        CREDIT_ASSIGN["CooperativeCreditAssigner"]
        SHAPED_REWARD["ShapedRewardManager"]
    end

    ADAPTIVE_RATE["Adaptive Learning Rate"]
    ELIGIBILITY["Eligibility Traces ( )"]
end

subgraph SKILLS_LAYER["  SKILLS SYSTEM"]
    SKILLS_REG["SkillsRegistry<br/>core/registry/skills_registry.py"]

    subgraph SKILL_DIRS["  Skill Sources"]
        REPO_SKILLS["/skills/"]
        CLAUDE_SKILLS["~/.claude/skills/"]
    end

    subgraph SKILL_STRUCT["  Skill Structure"]
        SKILL_MD["SKILL.md<br/>(metadata)"]
        TOOLS_PY["tools.py<br/>(callables)"]
    end

    subgraph SKILL_CATEGORIES["  Skill Categories"]
        INPUT_SKILLS["INPUT<br/>web-search, file-read"]
        OUTPUT_SKILLS["OUTPUT<br/>docx-tools, telegram"]
        TRANSFORM_SKILLS["TRANSFORM<br/>document-converter"]
    end
end

subgraph PROVIDER_LAYER["  LLM PROVIDER SYSTEM"]
    UNIFIED_PROVIDER["UnifiedLMProvider<br/>core/foundation/unified_lm_provider.py"]

    subgraph PROVIDERS["  Supported Providers"]
        ANTHROPIC["Anthropic"]
        OPENAI["OpenAI"]
        GOOGLE["Google"]
        GROQ["Groq"]
        OPENROUTER["OpenRouter"]
        CLAUDE_CLI["Claude CLI"]
        CURSOR_CLI["Cursor CLI"]
    end

    AUTO_DETECT["Auto-Detection<br/>Priority Chain"]
```

```
end

subgraph CONFIG_LAYER["  CONFIGURATION SYSTEM"]
    CLI_CONFIG["CLIConfig<br/>cli/config/schema.py"]
    JOTTY_CONFIG["JottyConfig<br/>(21 categories)"]
    AGENT_CONFIG["AgentConfig"]

    subgraph CONFIG_CATS["  Config Categories"]
        PERSIST_CFG["Persistence"]
        EXEC_CFG["Execution"]
        MEMORY_CFG["Memory"]
        LEARNING_CFG["Learning"]
        VALIDATION_CFG["Validation"]
    end
end

subgraph DATA_LAYER["  DATA STRUCTURES"]
    DATA_STRUCTS["core/foundation/data_structures.py"]
    EPISODE_RESULT["EpisodeResult"]
    AGENT_STATE["AgentState"]
    SWARM_STATE["SwarmState"]
    TASK_BOARD["SwarmTaskBoard"]
end

%% User interactions
USER --> CLI
CLI --> RENDERER
RENDERER --> USER

%% CLI to Commands
CLI --> CMD_REGISTRY
CMD_REGISTRY --> RUN_CMD
CMD_REGISTRY --> EXPORT_CMD
CMD_REGISTRY --> RESUME_CMD
CMD_REGISTRY --> JUSTJOT_CMD

%% Commands to Orchestration
RUN_CMD --> SWARM_MGR
JUSTJOT_CMD --> LEAN_EXEC

%% Orchestration internal
SWARM_MGR --> AGENT_RUNNER
SWARM_MGR --> LEAN_EXEC
SWARM_MGR --> SWARM_INSTALL
```

```
    SWARM_INSTALL --> TEMPLATE_REG
    TEMPLATE_REG --> SWARM_LEAN
    TEMPLATE_REG --> BASE_TEMPLATE

    %% Orchestration to Agents
    AGENT_RUNNER --> AUTO_AGENT
    AGENT_RUNNER --> INSPECTOR
    LEAN_EXEC --> AGENTIC_PLANNER

    %% Agent internal
    AUTO_AGENT --> AGENTIC_PLANNER
    AGENTIC_PLANNER --> TASK_INFERENCE
    AGENTIC_PLANNER --> WORKFLOW_PLAN

    %% Agents to Skills
    AUTO_AGENT --> SKILLS_REG
    LEAN_EXEC --> SKILLS_REG
    SKILLS_REG --> REPO_SKILLS
    SKILLS_REG --> CLAUDE_SKILLS
    REPO_SKILLS --> SKILL_MD
    REPO_SKILLS --> TOOLS_PY

    %% Memory connections
    SWARM_MGR --> CORTEX
    AGENT_RUNNER --> CORTEX
    CORTEX --> EPISODIC
    CORTEX --> SEMANTIC
    CORTEX --> PROCEDURAL
    CORTEX --> META
    CORTEX --> CAUSAL
    CORTEX --> LEVEL_CLASSIFIER
    CORTEX --> RAG_RETRIEVER
    CORTEX --> DEDUP_ENGINE
    CORTEX --> BRAIN_STATE
    BRAIN_STATE --> HIPPO_EXTRACT
    BRAIN_STATE --> RIPPLE_CONSOL
    CORTEX --> CAUSAL_EXTRACT

    %% Learning connections
    AGENT_RUNNER --> TD_LEARNER
    TD_LEARNER --> TRAJECTORY_PRED
    TD_LEARNER --> DIVERGENCE_MEM
    TD_LEARNER --> CREDIT_ASSIGN
    TD_LEARNER --> SHAPED_REWARD
    TD_LEARNER --> ADAPTIVE_RATE
    TD_LEARNER --> ELIGIBILITY

    %% Provider connections
    SWARM_MGR --> UNIFIED_PROVIDER
    AGENTIC_PLANNER --> UNIFIED_PROVIDER
    AUTO_AGENT --> UNIFIED_PROVIDER
    LEAN_EXEC --> UNIFIED_PROVIDER
    UNIFIED_PROVIDER --> ANTHROPIC
    UNIFIED_PROVIDER --> OPENAI
    UNIFIED_PROVIDER --> GOOGLE
    UNIFIED_PROVIDER --> GROQ
    UNIFIED_PROVIDER --> OPENROUTER
    UNIFIED_PROVIDER --> CLAUDE_CLI
    UNIFIED_PROVIDER --> CURSOR_CLI
    UNIFIED_PROVIDER --> AUTO_DETECT

    %% Config connections
    CLI --> CLI_CONFIG
    SWARM_MGR --> JOTTY_CONFIG
    AGENT_RUNNER --> AGENT_CONFIG
    CLI_CONFIG --> PERSIST_CFG
    CLI_CONFIG --> EXEC_CFG
    JOTTY_CONFIG --> MEMORY_CFG
    JOTTY_CONFIG --> LEARNING_CFG
    JOTTY_CONFIG --> VALIDATION_CFG

    %% Data structure connections
    SWARM_MGR --> EPISODE_RESULT
    AGENT_RUNNER --> AGENT_STATE
    SWARM_MGR --> SWARM_STATE
    SWARM_MGR --> TASK_BOARD

    classDef userLayer fill:#e1f5fe,stroke:#01579b
    classDef cmdLayer fill:#f3e5f5,stroke:#4a148c
    classDef orchLayer fill:#fff3e0,stroke:#e65100
    classDef agentLayer fill:#e8f5e9,stroke:#1b5e20
    classDef memLayer fill:#fce4ec,stroke:#880e4f
    classDef learnLayer fill:#f1f8e9,stroke:#33691e
    classDef skillLayer fill:#e0f2f1,stroke:#004d40
    classDef providerLayer fill:#e8eaf6,stroke:#1a237e
    classDef configLayer fill:#fff8e1,stroke:#ff6f00
    classDef dataLayer fill:#eceff1,stroke:#37474f
    class USER_LAYER userLayer
    class COMMAND_LAYER cmdLayer
    class ORCHESTRATION_LAYER orchLayer
    class AGENT_LAYER agentLayer
    class MEMORY_LAYER memLayer
    class LEARNING_LAYER learnLayer
    class SKILLS_LAYER skillLayer
    class PROVIDER_LAYER providerLayer
    class CONFIG_LAYER configLayer
    class DATA_LAYER dataLayer
```

Execution Flow Diagram

```
sequenceDiagram
    participant U as User
    participant CLI as JottyCLI
    participant CMD as CommandRegistry
    participant SM as SwarmManager
    participant AR as AgentRunner
    participant LE as LeanExecutor
    participant AP as AgenticPlanner
    participant AA as AutoAgent
    participant INS as InspectorAgent
    participant SR as SkillsRegistry
    participant MEM as MemoryCortex
    participant LRN as TDLambdaLearner
    participant LLM as UnifiedLMProvider
    participant RND as UIRenderer

    U->>CLI: Natural Language Query
    CLI->>CMD: Route Input

    alt Slash Command
        CMD->>SM: Execute Command
    else Natural Language
        CMD->>LE: Process Query
    end

    LE->>AP: Create Execution Plan
    AP->>LLM: Infer Task Type
    LLM-->>AP: RESEARCH|CREATION|etc.
    AP-->>LE: Workflow Plan
    LE->>SR: Discover Skills
    SR-->>LE: Available Skills

    LE->>SM: Execute with Swarm
    SM->>AR: Run Agent(s)

    loop Per Agent
        AR->>INS: Architect Validation (Pre)
        INS->>LLM: Planning Consensus
        LLM-->>INS: Approved Plan
        INS-->>AR: Validated Plan

        AR->>AA: Execute Task
        AA->>SR: Get Required Skills
        SR-->>AA: Skill Functions
        AA->>LLM: Generate Output
        LLM-->>AA: Result
        AA-->>AR: Agent Output

        AR->>INS: Auditor Validation (Post)
        INS->>LLM: Quality Check
        LLM-->>INS: Validation Result
        INS-->>AR: Final Approval

        AR->>LRN: Update Learning
        LRN->>LRN: TD( ) Update
        AR->>MEM: Store Experience
        MEM->>MEM: Classify & Store
    end

    AR-->>SM: Episode Result
    SM-->>CLI: Final Output
    CLI->>RND: Format Display
    RND-->>U: Rendered Result
```

Memory System Deep Dive

```
flowchart TB
    subgraph INPUT["  Memory Input"]
        EXP["Experience/Event"]
        QUERY["Query/Retrieval"]
    end
    subgraph CLASSIFICATION["  LLM Classification"]
        CLASSIFIER["MemoryLevelClassifier"]
        SALIENCE["HippocampalExtractor<br/>(Importance Score)"]
    end

    subgraph STORAGE["  5-Level Storage"]
        direction TB
        L1["EPISODIC<br/>       <br/>• Raw experiences<br/>• Fast decay (hours)<br/>• High c
        L2["SEMANTIC<br/>       <br/>• Patterns & abstractions<br/>• Slow decay (days)<br/>•
        L3["PROCEDURAL<br/>       <br/>• How-to knowledge<br/>• Medium decay<br/>• Action se
        L4["META<br/>       <br/>• Learning wisdom<br/>• No decay<br/>• Self-improvement"]
        L5["CAUSAL<br/>       <br/>• Why relationships<br/>• No decay<br/>• Cause-effect"]
    end

    subgraph RETRIEVAL["  Retrieval System"]
        RAG["LLMRAGRetriever"]
        DEDUP["DeduplicationEngine"]
        CONTEXT["Context Assembly"]
    end

    subgraph CONSOLIDATION["  Sleep Consolidation"]
        BSM["BrainStateMachine"]
        AWAKE["AWAKE Mode<br/>(Learning)"]
        SLEEP["SLEEP Mode<br/>(Consolidation)"]
        SWR["SharpWaveRipple<br/>Consolidator"]
        CAUSAL_EX["CausalExtractor"]
    end

    EXP --> CLASSIFIER
    CLASSIFIER --> SALIENCE
    SALIENCE --> L1
    L1 -.->|pattern extraction| L2
    L1 -.->|procedure extraction| L3
    L2 -.->|wisdom extraction| L4
    L1 -.->|causal extraction| L5

    QUERY --> RAG
    RAG --> DEDUP
    DEDUP --> L1
    DEDUP --> L2
    DEDUP --> L3
    DEDUP --> L4
    DEDUP --> L5
    L1 --> CONTEXT
    L2 --> CONTEXT
    L3 --> CONTEXT
    L4 --> CONTEXT
    L5 --> CONTEXT

    BSM --> AWAKE
    BSM --> SLEEP
    SLEEP --> SWR
    SWR --> L2
    SLEEP --> CAUSAL_EX
    CAUSAL_EX --> L5

    classDef input fill:#bbdefb,stroke:#1565c0
    classDef classify fill:#c8e6c9,stroke:#2e7d32
    classDef storage fill:#fff9c4,stroke:#f9a825
    classDef retrieve fill:#d1c4e9,stroke:#512da8
    classDef consolidate fill:#ffccbc,stroke:#bf360c

    class INPUT input
    class CLASSIFICATION classify
    class STORAGE storage
    class RETRIEVAL retrieve
    class CONSOLIDATION consolidate
```

Skills Execution Flow

```
flowchart LR
    subgraph TASK["  Task Input"]
        NL["Natural Language<br/>Task Description"]
    end

    subgraph PLANNING["  Planning Phase"]
        AP["AgenticPlanner"]
        INFER["Task Type<br/>Inference"]
        WORKFLOW["Workflow<br/>Generation"]
    end

    subgraph DISCOVERY["  Skill Discovery"]
        SR["SkillsRegistry"]
        SCAN["Lazy Scan<br/>Directories"]
        MATCH["Match Skills<br/>to Task"]
    end
    subgraph SKILL_SOURCES["  Skill Sources"]
        DIR1["/skills/<br/>  web-search/<br/>  docx-tools/<br/>  telegram-sender/<br/>  .
        DIR2["~/.claude/skills/<br/>  custom-skill/<br/>  ..."]
    end

    subgraph EXECUTION["  Execution"]
        STEP1["Step 1<br/>${input}"]
        STEP2["Step 2<br/>${step1_output}"]
        STEP3["Step 3<br/>${step2_output}"]
        AGG["Aggregate<br/>Results"]
    end

    subgraph OUTPUT["  Output"]
        FORMAT["Format Output<br/>(DOCX/PDF/MD/Telegram)"]
        RESULT["Final Result"]
    end

    NL --> AP
    AP --> INFER
    INFER --> WORKFLOW
    WORKFLOW --> SR
    SR --> SCAN
    SCAN --> DIR1
    SCAN --> DIR2
    SR --> MATCH
    MATCH --> STEP1
    STEP1 -->|"${output}"| STEP2
    STEP2 -->|"${output}"| STEP3
    STEP3 --> AGG
    AGG --> FORMAT
    FORMAT --> RESULT

    classDef task fill:#e3f2fd,stroke:#1565c0
    classDef plan fill:#f3e5f5,stroke:#7b1fa2
    classDef discover fill:#e8f5e9,stroke:#388e3c
    classDef source fill:#fff3e0,stroke:#ef6c00
    classDef exec fill:#e0f7fa,stroke:#00838f
    classDef output fill:#fce4ec,stroke:#c2185b

    class TASK task
    class PLANNING plan
    class DISCOVERY discover
    class SKILL_SOURCES source
    class EXECUTION exec
    class OUTPUT output
```

Learning System (TD-Lambda with A-Team)

```
flowchart TB
    subgraph EXPERIENCE["  Experience Collection"]
        STATE["State s_t"]
        ACTION["Action a_t"]
        REWARD["Reward r_t"]
        NEXT_STATE["State s_{t+1}"]
    end

    subgraph TD_CORE["  TD( ) Core"]
        TD_ERROR["TD Error<br/>  = r +  V(s') - V(s)"]
        ELIG["Eligibility Traces<br/>e(s) =  e(s) + 1"]
        VALUE_UPDATE["Value Update<br/>V(s) +=  · ·e(s)"]
    end

    subgraph ADAPTIVE["  Adaptive Components"]
        ALPHA["Adaptive  <br/>(Learning Rate)"]
        LAMBDA["  Parameter<br/>0=TD(0), 1=MC"]
        GAMMA["  Discount<br/>Factor"]
    end

    subgraph MARL["  Multi-Agent RL"]
        PRED["LLMTrajectory<br/>Predictor"]
        DIV_MEM["Divergence<br/>Memory"]
        CREDIT["Cooperative<br/>Credit Assigner"]
    end

    subgraph SHAPING["  Reward Shaping"]
        PER_STEP["Per-Step<br/>Rewards"]
        PER_AGENT["Per-Agent<br/>Rewards"]
        COOP["Cooperative<br/>Bonuses"]
        SHAPED["ShapedReward<br/>Manager"]
    end

    subgraph OUTPUT["  Learning Output"]
        Q_VALUES["Updated<br/>Q-Values"]
        POLICY["Improved<br/>Policy"]
        MEMORY["Experience<br/>Storage"]
    end
    STATE --> TD_ERROR
    ACTION --> TD_ERROR
    REWARD --> TD_ERROR
    NEXT_STATE --> TD_ERROR

    TD_ERROR --> ELIG
    ELIG --> VALUE_UPDATE

    ALPHA --> VALUE_UPDATE
    LAMBDA --> ELIG
    GAMMA --> TD_ERROR

    ACTION --> PRED
    PRED --> DIV_MEM
    DIV_MEM -->|"Learn why diverge"| CREDIT
    CREDIT --> COOP

    PER_STEP --> SHAPED
    PER_AGENT --> SHAPED
    COOP --> SHAPED
    SHAPED --> REWARD

    VALUE_UPDATE --> Q_VALUES
    Q_VALUES --> POLICY
    TD_ERROR --> MEMORY

    classDef exp fill:#e8eaf6,stroke:#3949ab
    classDef core fill:#fff8e1,stroke:#ffa000
    classDef adaptive fill:#e0f2f1,stroke:#00897b
    classDef marl fill:#fce4ec,stroke:#d81b60
    classDef shape fill:#f3e5f5,stroke:#8e24aa
    classDef output fill:#e8f5e9,stroke:#43a047

    class EXPERIENCE exp
    class TD_CORE core
    class ADAPTIVE adaptive
    class MARL marl
    class SHAPING shape
    class OUTPUT output
```

Provider Auto-Detection Chain

```
flowchart LR
    subgraph DETECT["  Auto-Detection"]
        START["Start"]
        CHECK1{"Claude CLI<br/>Available?"}
        CHECK2{"Cursor CLI<br/>Available?"}
        CHECK3{"ANTHROPIC_API_KEY<br/>Set?"}
        CHECK4{"OPENROUTER_API_KEY<br/>Set?"}
        CHECK5{"OPENAI_API_KEY<br/>Set?"}
        CHECK6{"GROQ_API_KEY<br/>Set?"}
    end

    subgraph PROVIDERS["  Providers"]
        CLAUDE["Claude CLI<br/>(Free, Local)"]
        CURSOR["Cursor CLI<br/>(Free, IDE)"]
        ANTHRO["Anthropic API<br/>(claude-3-5-sonnet)"]
        OPENR["OpenRouter<br/>(Multi-model)"]
        OPENAI["OpenAI API<br/>(gpt-4)"]
        GROQ["Groq API<br/>(Fast inference)"]
        FAIL["No Provider<br/>Available"]
    end

    START --> CHECK1
    CHECK1 -->|Yes| CLAUDE
    CHECK1 -->|No| CHECK2
    CHECK2 -->|Yes| CURSOR
    CHECK2 -->|No| CHECK3
    CHECK3 -->|Yes| ANTHRO
    CHECK3 -->|No| CHECK4
    CHECK4 -->|Yes| OPENR
    CHECK4 -->|No| CHECK5
    CHECK5 -->|Yes| OPENAI
    CHECK5 -->|No| CHECK6
    CHECK6 -->|Yes| GROQ
    CHECK6 -->|No| FAIL

    classDef detect fill:#e3f2fd,stroke:#1976d2
    classDef provider fill:#e8f5e9,stroke:#388e3c
    classDef fail fill:#ffebee,stroke:#c62828

    class DETECT detect
    class PROVIDERS provider
    class FAIL fail
```

Complete Component Interaction Matrix

```
graph TB
    subgraph LAYER1["Layer 1: User Interface"]
        A1["JottyCLI"]
        A2["UIRenderer"]
    end

    subgraph LAYER2["Layer 2: Commands"]
        B1["CommandRegistry"]
        B2["run/export/resume"]
    end

    subgraph LAYER3["Layer 3: Orchestration"]
        C1["SwarmManager"]
        C2["AgentRunner"]
        C3["LeanExecutor"]
    end

    subgraph LAYER4["Layer 4: Agents"]
        D1["AgenticPlanner"]
        D2["AutoAgent"]
        D3["InspectorAgent"]
    end

    subgraph LAYER5["Layer 5: Core Systems"]
        E1["SkillsRegistry"]
        E2["MemoryCortex"]
        E3["TDLambdaLearner"]
    end

    subgraph LAYER6["Layer 6: Foundation"]
        F1["UnifiedLMProvider"]
        F2["DataStructures"]
        F3["Configuration"]
    end

    A1 <--> A2
    A1 --> B1
    B1 --> B2
    B2 --> C1
    B2 --> C3
    C1 <--> C2
    C1 <--> C3
    C2 --> D1
    C2 --> D2
    C2 --> D3
    C3 --> D1
    D1 <--> D2
    D2 --> E1
    C2 --> E2
    C2 --> E3
    D1 --> F1
    D2 --> F1
    D3 --> F1
    E1 --> F2
    E2 --> F2
    E3 --> F2
    C1 --> F3
    C2 --> F3

    classDef layer1 fill:#e1f5fe,stroke:#0288d1
    classDef layer2 fill:#f3e5f5,stroke:#7b1fa2
    classDef layer3 fill:#fff3e0,stroke:#ff8f00
    classDef layer4 fill:#e8f5e9,stroke:#388e3c
    classDef layer5 fill:#fce4ec,stroke:#c2185b
    classDef layer6 fill:#eceff1,stroke:#546e7a

    class LAYER1 layer1
    class LAYER2 layer2
    class LAYER3 layer3
    class LAYER4 layer4
    class LAYER5 layer5
    class LAYER6 layer6
```

File Structure Reference

```
Jotty/
  cli/
      app.py                   # JottyCLI - Main entry point
      ui/
          renderer.py          # UIRenderer - Display formatting
      commands/
          __init__.py          # CommandRegistry
          run.py               # RunCommand
          export.py            # ExportCommand
```

```
        resume.py           # ResumeCommand
        justjot.py          # JustJotCommand
    config/
        schema.py           # CLIConfig

core/
    orchestration/
        v2/
            swarm_manager.py  # SwarmManager - Central orchestrator
            agent_runner.py   # AgentRunner - Per-agent execution
            lean_executor.py  # LeanExecutor - LLM-first execution
            swarm_installer.py # SwarmInstaller
            templates/
                __init__.py   # TemplateRegistry
                registry.py   # Template registration
                swarm_lean.py # SwarmLean template

    agents/
        agentic_planner.py   # AgenticPlanner - LLM-based planning
        auto_agent.py        # AutoAgent - Autonomous execution

    memory/
        cortex.py            # MemoryCortex - Brain-inspired memory

    registry/
        skills_registry.py   # SkillsRegistry - Skill management

    foundation/
        data_structures.py   # Core types & dataclasses
        unified_lm_provider.py # LLM provider abstraction

skills/
    web-search/              # Web search skill
    docx-tools/              # Document generation
    telegram-sender/         # Telegram integration
    financial-visualization/ # Charts & graphs
    ...                      # 25+ more skills
```

Design Patterns Used

Pattern	Component	Purpose
Registry	SkillsRegistry, TemplateRegistry, CommandRegistry	Central lookup for components

Pattern	Component	Purpose
Strategy	Validators, Memory Classifiers	Interchangeable algorithms
Adapter	UnifiedLMProvider	Abstract different LLM APIs
Factory	LM Provider creation	Create appropriate provider
Observer	Status callbacks	Progress notifications
Lazy Init	Skills, SwarmManager components	Defer expensive initialization
Chain of Responsibility	Command execution	Handle commands in sequence
Composite	Multi-level memory	Tree structure for memories
State	BrainStateMachine	Manage AWAKE/SLEEP transitions
Template Method	BaseTemplate, SwarmLean	Define algorithm skeleton

---

Key Architectural Principles

1. **LLM-First Design**: All decisions via LLM, no hardcoded rules
2. **Zero-Configuration Intelligence**: Natural language → automatic agent creation
3. **Brain-Inspired Memory**: 5-level hierarchy with consolidation
4. **Temporal-Difference Learning**: TD( ) with adaptive rates
5. **Skill-Based Autonomy**: Modular, discoverable capabilities
6. **Lazy Initialization**: Fast startup, load on demand
7. **Multi-Provider Abstraction**: Unified interface, auto-detection
8. **Separation of Concerns**: Clear layer boundaries