

Contents

Reference Architecture: The Reflective Adaptive Agent	1
Overview	1
Perché Questa Architettura	2
Decision Rationale	2
Applicabilità	2
Architettura ad Alto Livello	2
Componenti Core (8)	3
1. Goal Analysis	3
2. Planning Engine	3
3. Execution Engine	3
4. Reflection Module	3
5. Memory System	3
6. Tool Registry	3
7. Model Router	3
8. Safety Verifier	4
Bounded Emergence Implementation	4
Struttura Documenti	4
Performance Caratteristiche	4
Target Metrics	4
Scalability	4
Evoluzione Path	4
Comparison con Alternative	5
Getting Started	5
Per Architetti	5
Per Implementatori	5
Per Ricercatori	6

Reference Architecture: The Reflective Adaptive Agent

Architettura di riferimento che rappresenta l’optimal balance tra i trade-off identificati nel framework di ricerca.

Overview

Questa è l’architettura più vicina all’“ideale” per la **maggioranza dei casi d’uso** (circa 70-80%), basata su:

- Analysis sistematica dei trade-off (vedere /02-design-dimensions.md)
- Pattern 2 “Reflective Agent” (vedere /03-architecture-patterns.md)
- Bounded Emergence principles (vedere /05-bounded-emergence.md)
- Validazione teorica contro problem taxonomy

Perché Questa Architettura

Decision Rationale

Problem Class Target: Classe B (Complex Planning) con adaptability a Classe A

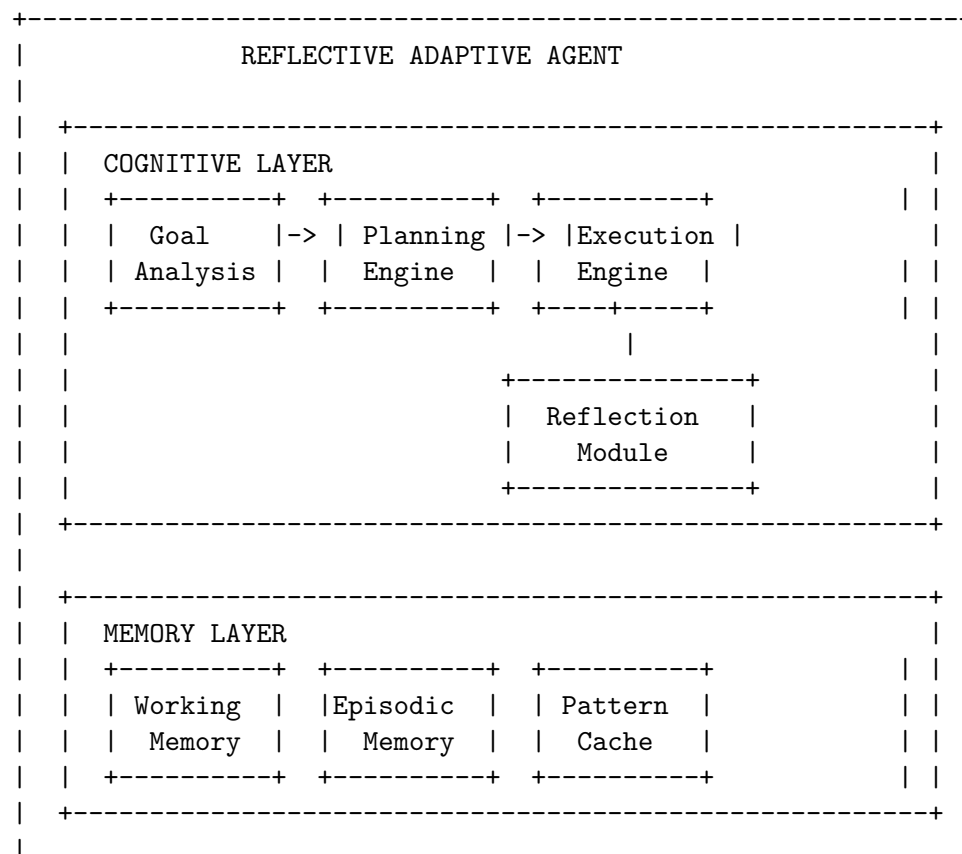
Design Dimensions Scelte: - **Controllo vs Flessibilità:** Hybrid (60% emergenza, 40% controllo) - **Semplicità vs Completeness:** Balanced (8 componenti core, estensibile) - **Generalità vs Specializzazione:** General-purpose con specialization capability - **Latenza vs Qualità:** Responsive (target 5-30s per task medio) - **Cost vs Capability:** Optimized (routing intelligente modelli) - **Autonomy vs Human:** Autonomous con fallback gates

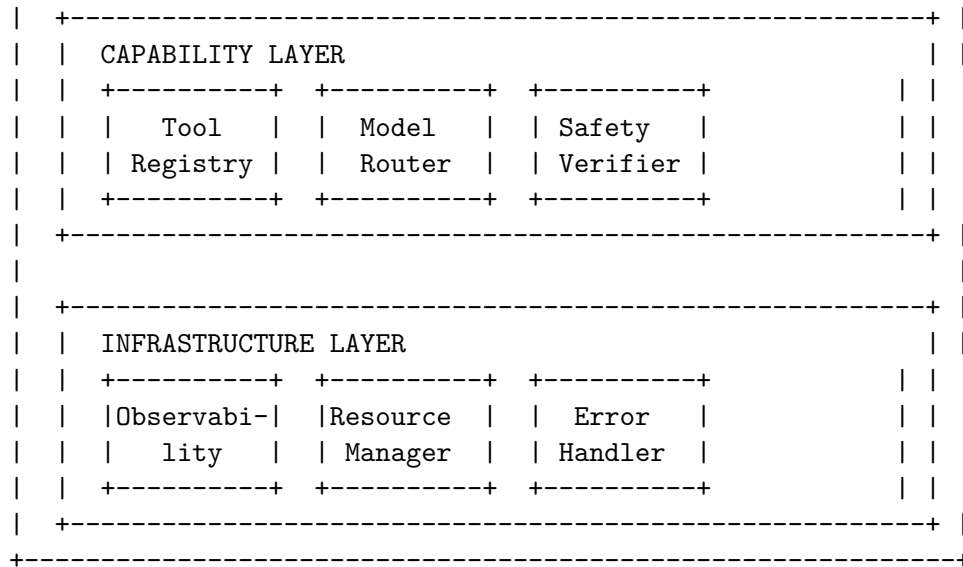
Applicabilità

[OK] **Ideale per:** - Software development assistants - Research and analysis automation - Complex problem solving - Data processing workflows - Content creation pipelines - Business process automation

[!] **Meno adatto per:** - Safety-critical systems (serve Verified Agent) - Real-time control (<100ms, serve Reactive Agent) - Simple Q&A (overkill, serve Minimal Loop) - Ultra-specialized domains (serve customizzazione)

Architettura ad Alto Livello





Componenti Core (8)

1. Goal Analysis

Analizza task ricevuto e identifica obiettivi, vincoli, success criteria.

2. Planning Engine

Genera execution plan con decomposizione gerarchica e gestione dipendenze.

3. Execution Engine

Esegue plan con monitoring, verifica, adaptation dinamica.

4. Reflection Module

Analizza episodi completati, estrae pattern, migliora strategie.

5. Memory System

Working (context), Episodic (history), Pattern Cache (learned strategies).

6. Tool Registry

Gestisce discovery, binding, execution sicura di capabilities esterne.

7. Model Router

Routing intelligente tra modelli (small/fast vs large/capable).

8. Safety Verifier

Multi-layer validation (input, output, actions) + bounded emergence enforcement.

Bounded Emergence Implementation

Emergent Behaviors: - Plan generation e adaptation - Strategy selection - Error recovery approaches - Pattern recognition e generalization

Bounds Enforced: - Input validation (schema, injection detection) - Tool whitelist con permissions - Output verification (format, constraints) - Safety bounds (prohibited actions) - Resource limits (token budget, time, cost)

Struttura Documenti

1. **01-system-architecture.md** - Diagrammi completi di sistema
2. **02-cognitive-layer.md** - Goal, Planning, Execution, Reflection
3. **03-memory-system.md** - Working, Episodic, Pattern Cache
4. **04-capability-layer.md** - Tools, Models, Safety
5. **05-infrastructure.md** - Observability, Resources, Errors
6. **06-data-flows.md** - Interaction patterns e sequenze
7. **07-decision-rationale.md** - Perché ogni scelta
8. **08-deployment.md** - Considerazioni operative

Performance Caratteristiche

Target Metrics

Metrica	Target	Razionale
Latency (Simple)	<5s	User tolerance per query semplici
Latency (Complex)	10-60s	Planning overhead accettabile
Success Rate	85-95%	High reliability senza over-engineering
Cost per Task	\$0.05-0.30	Model routing optimization
Adaptability	80%+	Generalizza a nuovi task types
Recovery Rate	90%+	Errori gestiti automaticamente

Scalability

- **Vertical:** Single instance gestisce 10-50 concurrent users
- **Horizontal:** Stateless design permette scaling lineare
- **Storage:** Memory system scala a milioni di episodi

Evoluzione Path

MVP (Week 1)

+ - Core execution loop

+ - Basic planning

- + - 5-10 essential tools
- + - Simple memory

Production (Month 1)

- + - MVP +
- + - Reflection module
- + - Pattern caching
- + - Model routing
- + - Full observability

Optimized (Month 3)

- + - Production +
- + - Advanced planning strategies
- + - Learned optimizations
- + - Domain customizations
- + - Multi-agent coordination (if needed)

Comparison con Alternative

Aspect	Minimal Loop	Reference Arch	Verified Agent
Complexity	Low	Medium	High
Success Rate	70-85%	85-95%	95-99%
Latency	<2s	5-30s	Variable
Cost	\$0.01-0.05	\$0.05-0.30	\$0.50-5.00
Flexibility	Very High	High	Low
Safety	Low	Medium-High	Very High

Reference Architecture offre **optimal balance** per maggioranza use cases.

Getting Started

Per Architetti

1. Leggere 07-decision-rationale.md per capire trade-off
2. Studiare 01-system-architecture.md per overview
3. Approfondire componenti specifici di interesse

Per Implementatori

1. Iniziare con 02-cognitive-layer.md (core)
2. Implementare layer by layer
3. Usare 06-data-flows.md per interaction patterns
4. Riferirsi a 08-deployment.md per operations

Per Ricercatori

1. Analizzare 07-decision-rationale.md per design choices
2. Identificare aree per optimization
3. Proporre modifiche basate su use case specific

Next: 01-system-architecture.md -> Deep dive nell'architettura completa