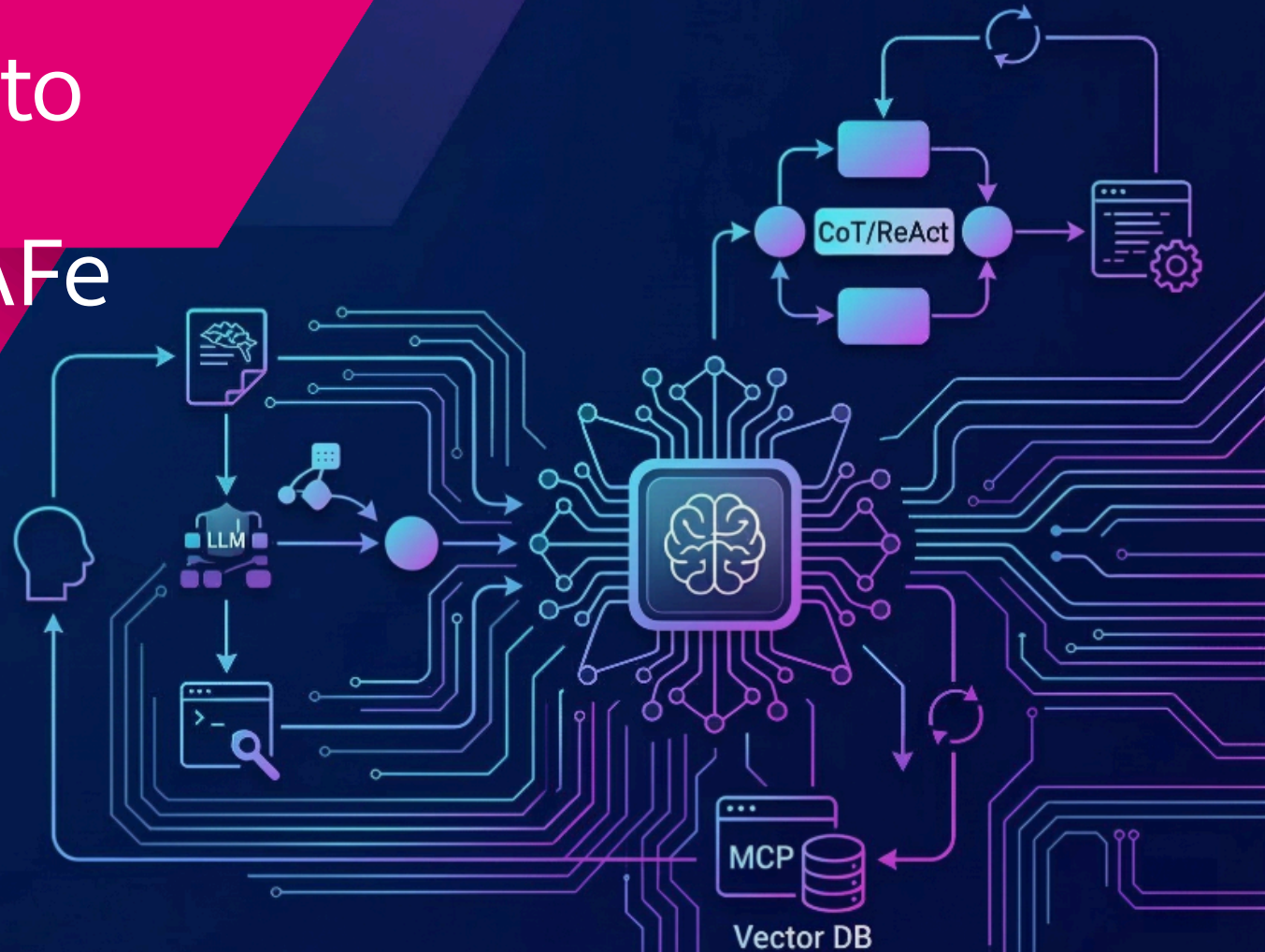


From Architecture to Multi-Agent Collaboration in SAFe

A Practical Guide to Building Intelligent Workflows

Ding, Shuo | March 2026



Our holistic portfolio goes far beyond a single AI use case

Transform your business and gain a competitive edge by using AI

- AI Use Case Prioritization & Development
- AI Maturity & Empowerment
- AI Act Readiness and AI Ethics
- AI Culture and Change



Research & Development



Production



Logistic



Sales



Operation & Service

IT Foundation AI Cases

Corporate Function AI Cases

AI Industry Scenarios

IT Solutions

IT Service Deck Automation	Employee Chatbot
AI automated Coding	IT Resource Management
AI driven Legacy Transformation	Safety Monitoring, Observability

Horizontal Solutions

Human Resource	Resume screening HR Chatbot	Finance & Contr.	Risk Assessment Forecasting
Sales & Marketing	Virtual Assistant Person. Marketing	Legal	Law Monitoring Contract analysis
Aftersales & Service	Dynamic pricing Call Center Interaction	Security	Detection & Response AI Pen Testing

Industry Solutions

	Product Design Virtual Prototyping		Content Recognition Supply Chain Optimization
	Quality assurance Pred. maintenance		Smart Assembly Planning Reliability
	Customer Support & Services Lead scoring and follow up		

AI is making us more efficient

AI solutions apply across all horizontal functions

Finally, we focus on creating business value with



APPLICATIONS

MODELS

INFRASTRUCTURE

CHIPS

ENERGY



CHATBOTS



DIGITAL BIOLOGY



ROBOTAXI



ENTERPRISE AI AGENTS



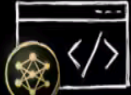
SCIENCE



ROBOTICS



MANUFACTURING



AI CODER

LLM

VLM

VLA

MMLLM

GPT

DM

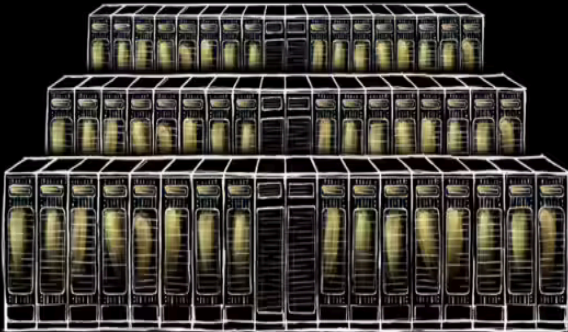
GNN

MOE

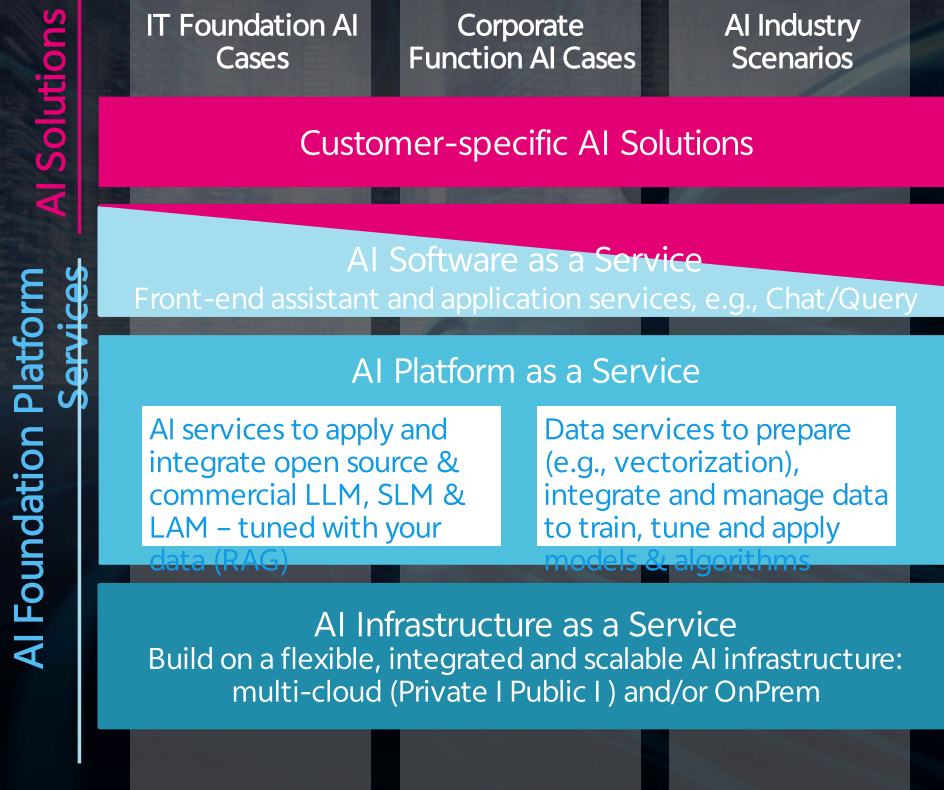
SSM

LBM

AI FACTORIES



Your Path towards successful AI application



Frontend Services

Smart Chat Interface

Content Recognition

Customer Segmentation

AI Automated Tasks

Add More...



LLM Hub API

Shared Instance

- Standardized APIs for access to the latest LLMs
- Shared resources
- Cost-effective pay-as-you-go pricing

Private Instance

- Hosting of finely tuned private models
- Dedicated resources
- Monthly fees



RAG API

Private Instance

- End-to-end hosting of the company document base
- Full RAG system access via APIs
- Customizing and automatic evaluation of over diverse retrieval settings
- 24/7 support
- Monthly fees



AI Platform as a Service

Open-source LLMs + On-Prem

Proprietary LLMs + 3rd Party



AI Platform as a Service
Build on a flexible, integrated and scalable AI infrastructure: multi-cloud (Private | Public |) and/or OnPrem

* Large Language Models (LLM)

* Retrieval Augmented Generation (RAG)

Current Challenges in Enterprise R&D

The "Last Mile" Gap Between LLMs and DevOps Toolchains



Powerful LLM

"Brain in a Vat" - Passive Interaction



THE "LAST MILE" GAP



Complex DevOps Toolchain

"Information Silos" - Manual Context

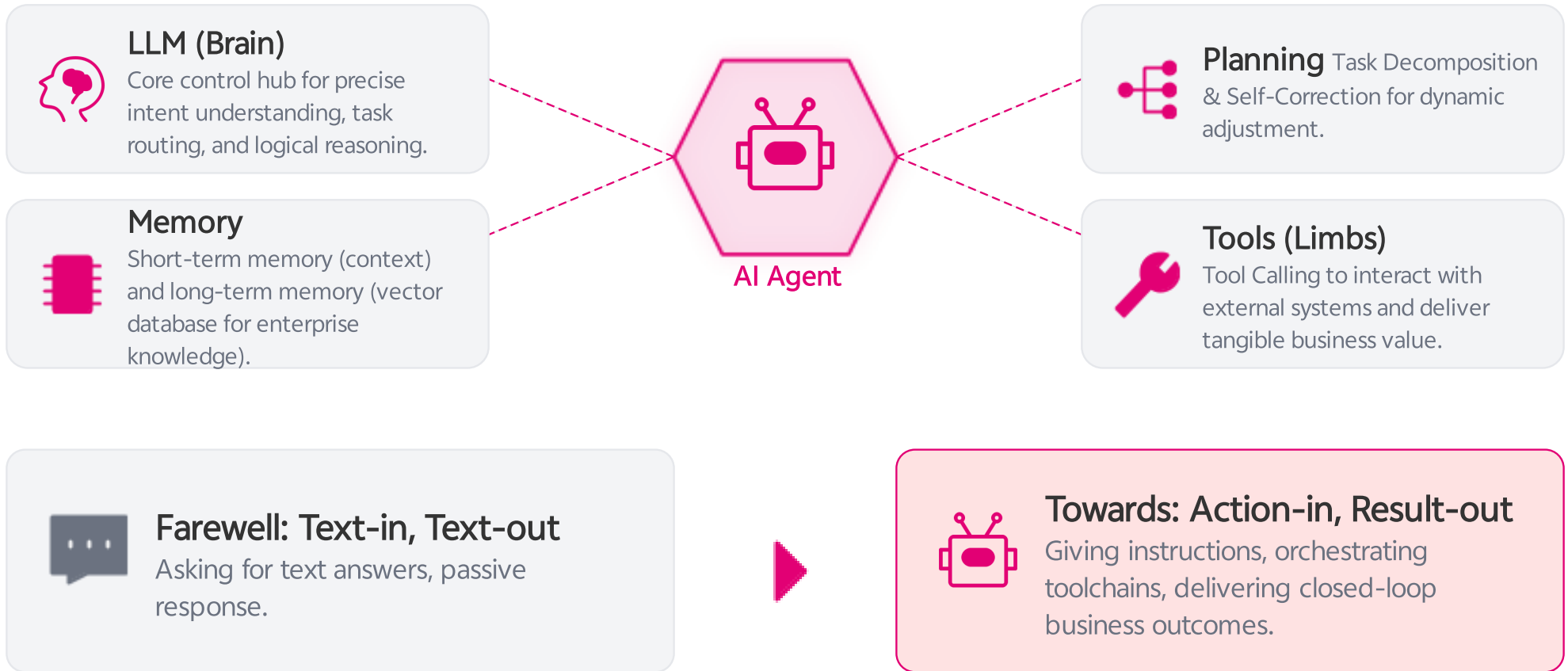


AI Agent: The Bridge & Executor

Equips LLMs with "hands and feet", enabling active connection and operation of DevOps tools (Jira, GitLab, CI/CD). Bridges the gap, automates workflows, and achieves autonomous closed-loop R&D tasks.

Defining the AI Agent: An Engineering Perspective

LLM + Memory + Planning + Tool Execution



The Evolution of LLM Applications

From Passive Chatbots to Autonomous Agents



Level 4: Multi-Agent

Multiple specialized 'Expert Agents' collaborate autonomously to deliver complex software, like teams in SAFe.



Level 3: Reasoning & Tooling

Incorporates Chain of Thought (CoT) and Tool Calling to perform actions and engage with enterprise data/workflows.



Level 2: Contextual

Introduces context window and short-term memory for coherent communication. A passive assistance tool.



Level 1: Chatbot

Single-turn dialogue, no memory, compresses and retrieves vast knowledge. Limited utility.



Paradigm Shift

From “Management Tools” to
“Expert Management Teams”



Core Building Blocks of Agentic Systems

Tool Calling, Skills, MCP, ReAct, and CoT

The Limbs

Tool Calling



Equips the model with 'hands and feet'. Calls external functions via structured output (e.g., JSON) to perform actions.

MCP (Model Context Protocol)

The 'common language' for unified access. Defines a standard protocol for zero-cost integration with enterprise systems.

Skill

Encapsulates complex APIs into atomic skills. Describes skill usage in natural language to boost tool calling accuracy.

The Brain

CoT (Chain of Thought)

Forces the model to 'show its work'. Enhances reliability in complex reasoning and reduces hallucinations.

ReAct (Reason + Act)

A closed-loop control flow paradigm. Creates an infinite loop of Thought -> Action -> Observation for true autonomy.



Agent Construction = Connectivity (Tooling / MCP) + Skill Abstraction + Reasoning (CoT) + Control Flow (ReAct)

Mechanism Deep Dive: Tool Calling & MCP

Enabling LLMs to Interact with External Systems

Tool Calling: Protocol & Parameter Extraction

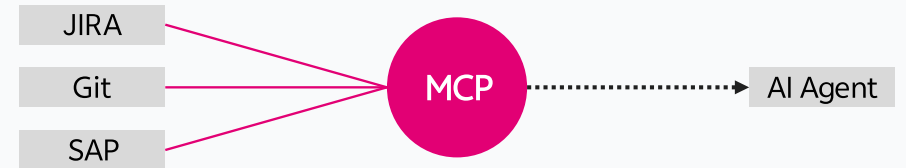
"LLMs only generate JSON, they do NOT execute code."



- No Direct Execution: LLMs only generate intent
- Structured Protocol: Parameters via JSON Schema
- The Loop: Model Decides → We Execute → Return Result

MCP: The "Common Language" for Integration

"Solves the scaling problem of heterogeneous systems."



- Standardized Language: One protocol, many sources
- Kill the "Glue Code": Zero-cost integration
- Interoperability: Model-agnostic enterprise data



Tool Calling: Solves "Can it act?" - Enables LLMs to output standard instructions.



MCP: Solves "Can it scale?" - Unifies heterogeneous enterprise systems.

Mechanism Deep Dive: Tool Calling & MCP

Sample

Request (Agent -> LLM)

```
1 {
2   "model": "gpt-4o",
3   "messages": [
4     {
5       "role": "user",
6       "content": "帮我看看昨天分配给我的那个 Jira Epic (EPIC-1024) 还有几个阻碍项?"
7     }
8   ],
9   "tools": [
10    {
11     "type": "function",
12     "function": {
13       "name": "get_epic_blockers",
14       "description": "获取指定 Jira Epic 的阻碍项 (Blockers) 列表及状态",
15       "parameters": {
16         "type": "object",
17         "properties": {
18           "epic_id": {
19             "type": "string",
20             "description": "Jira Epic 的 ID, 例如 EPIC-1024"
21           }
22         }
23       },
24       "required": ["epic_id"]
25     }
26   }
27 ],
28 "tool_choice": "auto"
29 }
```

Tool Info

Request (Agent -> LLM):

```
{
  "model": "gpt-4o",
  "messages": [
    {
      "role": "user",
      "content": "帮我看看昨天分配给我的那个 Jira Epic (EPIC-1024) 还有几个阻碍项?"
    },
    {
      "role": "assistant",
      "tool_calls": [
        {
          "id": "call_abc123",
          "type": "function",
          "function": {
            "name": "get_epic_blockers",
            "arguments": "{\"epic_id\":\"EPIC-1024\"}"
          }
        }
      ]
    },
    {
      "role": "tool",
      "tool_call_id": "call_abc123",
      "name": "get_epic_blockers",
      "content": "{\"status\": \"success\", \"blockers\": [{\"id\": \"BUG-501\", \"status\": \"In Progress\", \"assignee\": \"张三\"}]}"
    }
  ]
}
```

Tool Selection

Mechanism Deep Dive: Tool Calling & MCP

Sample

MCP Server

```
# mcp_jira_server.py
from mcp.server.fastmcp import FastMCP

# 初始化一个 MCP Server
mcp = FastMCP("Jira-R&D-Server")

@mcp.tool()
def get_epic_blockers(epic_id: str) -> str:
    """
    获取指定 Jira Epic 的阻碍项列表。

    Args:
        epic_id: Jira Epic 的 ID, 例如 EPIC-1024
    """
    # 这里是模拟调用企业内部真实的 Jira API
    print(f"[Server] 正在查询内部数据库, Epic: {epic_id}")

    # 模拟真实返回的数据
    mock_db = {
        "EPIC-1024": '{"blockers": ["BUG-501 (In Progress)"]}',
        "EPIC-1025": '{"blockers": []}'
    }

    return mock_db.get(epic_id, '{"error": "Epic not found"}')

if __name__ == "__main__":
    # 以 stdio 模式启动 (标准输入/输出), 这是 MCP 默认的跨进程通信方式
    mcp.run()
```

Function
Description

Request (MCP Client/Agent -> MCP Server)

```
1 {~
2   "jsonrpc": "2.0",~
3   "id": 1,~
4   "method": "tools/call",~
5   "params": {~
6     "name":
7     "get_epic_blockers",~
8     "arguments": {~
9       "epic_id": "EPIC-1024"~
10    }~
11 }
```

Response

```
1 {~
2   "jsonrpc": "2.0",~
3   "id": 1,~
4   "result": {~
5     "content": [~
6       {~
7         "type": "text",~
8         "text": "{\\"status\\":
9         \\"success\\", \\"blockers\\":
10        [\\"id\\": \\"BUG-501\\",
11        \\"status\\": \\"In
12        Progress\\"]}"~
13 }
```

Mechanism Deep Dive: ReAct & Chain of Thought

Handling Complex Tasks through Observation and Reasoning

CoT: Let's Think Step by Step

Forces the model to 'show its work', reducing hallucinations.

Step 1: Analyze Requirements

Step 2: Decompose Tasks

Step 3: Logical Validation

Objective: Eliminate Hallucination via "Slow Thinking"

ReAct: The "Think-Act-Observe" Loop

Enables dynamic planning, fault tolerance, and self-healing.



Thought

What next?



Action

Call tools



Obs

Real result



Self-Healing Example:

Failed Git pull → Observation: 'Branch not found' → Next Thought: 'Verify Jira ticket'.

Core Summary: The "Mind" of the Agent

- ✓ CoT: Teaches the model to deliberate and avoid nonsense.
- ✓ ReAct: Equips the model with 'dynamic correction' capabilities for real-world uncertainty.
- ✓ Together: Form the 'mind' of the Agent, enabling handling of unstructured, enterprise-grade tasks.

Mechanism Deep Dive: ReAct & Chain of Thought

Sample

System Prompt

```
1 你是一个智能研发助手。请你务必严格按照以下格式进行思考和执行任务：→
2 →
3 Thought: 你对当前情况的思考过程（你需要做什么）→
4 Action: 你决定调用的工具名称，必须是提供的工具之一→
5 Action Input: 工具调用的参数（JSON 格式）→
6 Observation: 工具执行返回的实际结果（由系统提供，你不要自己编造）
7 →
8 ...（重复上述过程，直到你得出最终结论）...→
9 →
10 Thought: 我现在知道了最终答案→
11 Final Answer: 返回给用户的最终回答
```

LLM Response (Thought + Action):

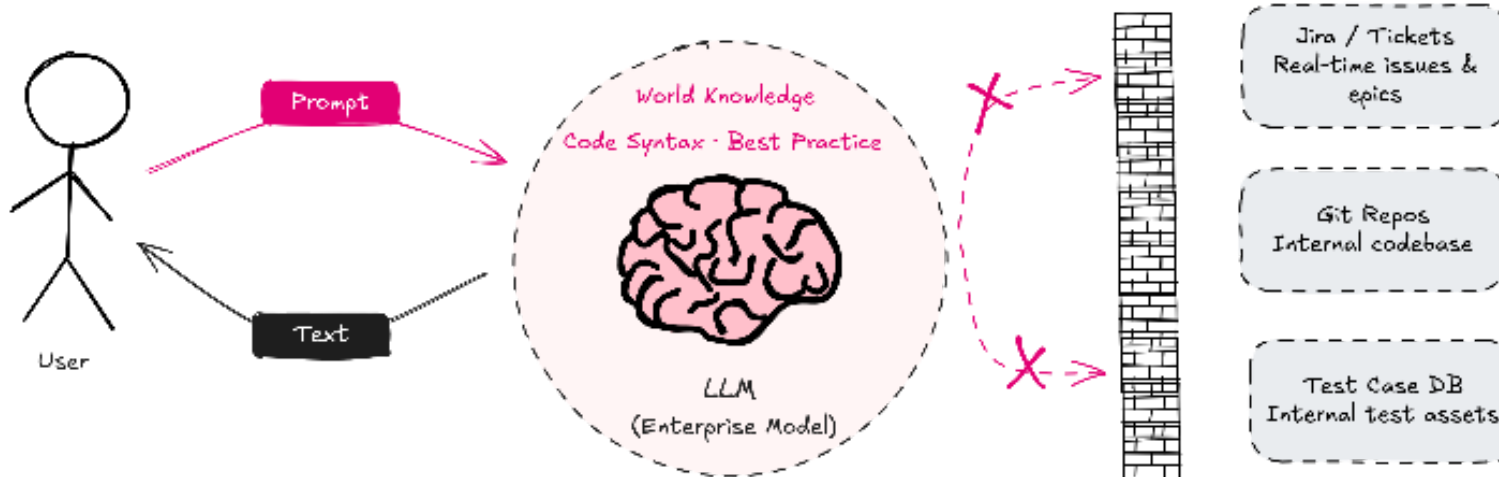
```
1 {
2   "role": "assistant",
3   "content": "Thought: 首先，我需要查找 EPTC-1024 的故障排除表，看看里面是否包含了 Bug。",
4   "tool_calls": [
5     {
6       "id": "call_1",
7       "function": {
8         "name": "get_epic_blockers",
9         "arguments": "{\"topic_id\":\"EPTC-1024\"}"
10      }
11    ]
12  }
13 }
14 →
15 {
16   "role": "tool",
17   "tool_call_id": "call_1",
18   "content": "{\"blockers\": [{\"id\": \"BUG-501\", \"title\": \"车载内存库内存泄漏\"}]}"
19 }
20 →
21 {
22   "role": "assistant",
23   "content": "Thought: 观察到 EPTC-1024 存在一个链接到 BUG-501（车载内存库内存泄漏）。根据用户的需求，我接下来需要去查阅 Wiki 来寻找关于这个 BUG-501 的解决方案。",
24   "tool_calls": [
25     {
26       "id": "call_2",
27       "function": {
28         "name": "search_confluence",
29         "arguments": "{\"query\":\"BUG-501 内存泄漏 解决方案\"}"
30      }
31    ]
32  }
33 }
34 →
35 {
36   "role": "tool",
37   "tool_call_id": "call_2",
38   "content": "{\"results\": [\"Wiki 文章: 内存泄漏解决方案 - 禁用内存泄漏检测可缓解 BUG-501 内存泄漏问题.\"]}"
39 }
40 →
41 {
42   "role": "assistant",
43   "content": "Thought: 我已经找到了问题及其解决方案，现在可以告诉用户了。关于 EPTC-1024 目前版本 *BUG-501（车载内存库内存泄漏）* 问题，我在内部 Wiki 为你找到了一个可行的解决方案：你可以尝试 *禁用内存泄漏检测* 来缓解这个内存泄漏问题。需要我帮你联系相关负责人吗？"
44 }
```

Architecture V1.0: The Base LLM





Text-in, Text-out (Isolated Context)

V1.0

TEXT IN · TEXT OUT — Minimal Loop



KEY OBSERVATIONS

-  Text-in, Text-out
Pure linguistic interface. No system access, no state awareness.
-  World knowledge ≠ Business context
Knows coding patterns. Knows nothing about your codebase, tickets, or users.
-  Hard isolation wall
Jira · Git · CI/CD · Test DB — all invisible. Every enterprise data source is unreachable.
-  "I cannot access real-time data."
The classic dead-end response when asked anything context-specific.

The "Blind Men & Elephant" Problem
High IQ, zero situational awareness.
A brilliant consultant locked in a black box — useful for generic advice, useless for real pipelines.
→ V2.0: Break the wall. Add hands & feet.

● V1.0 capability: language only

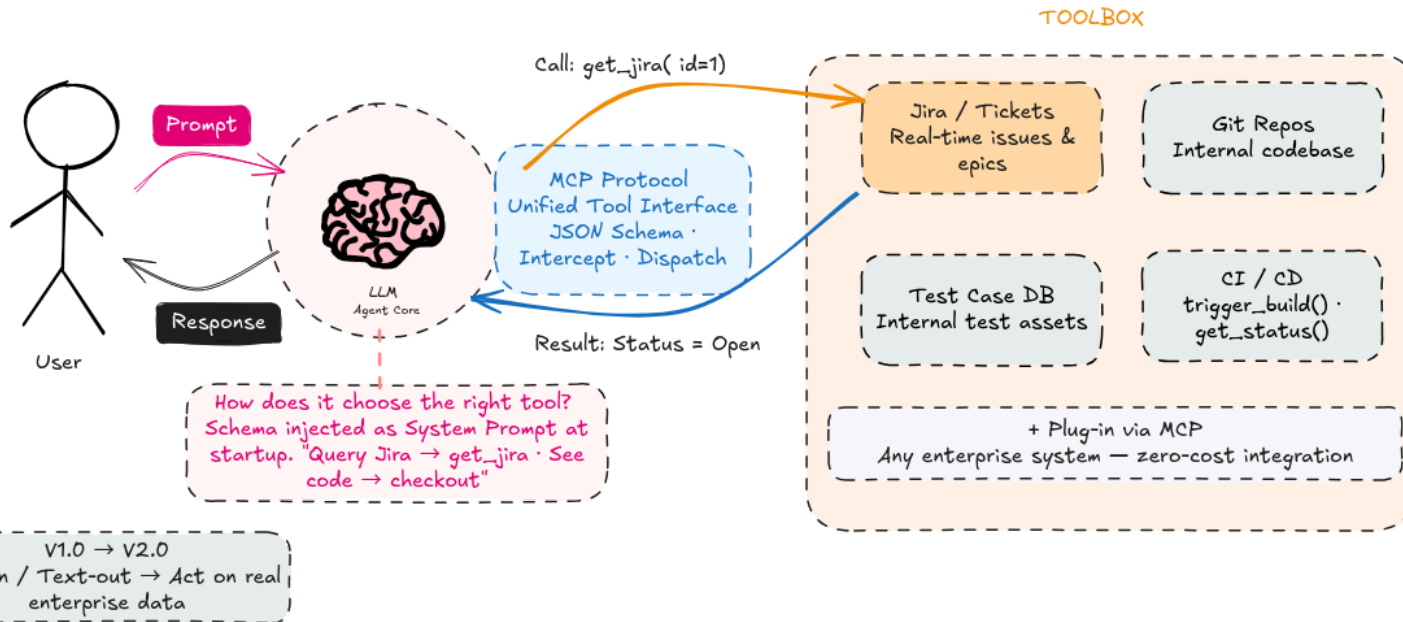
● Missing: real-time data, identity, system access

● Next: Tool Calling breaks the isolation

Architecture V2.0: Integrating Tool Calling & MCP

Establishing Bidirectional Data Flow with APIs

V2.0



KEY OBSERVATIONS

- Tool Calling** — not "hacking in" LLM outputs structured JSON commands. The Agent Engine intercepts & dispatches them to real APIs.
- Schema as "Operations Manual"** Tool definitions (name, params, description) injected as system prompt. LLM reads & routes precisely.
- MCP = Universal Socket** Like standardizing on REST/gRPC — heterogeneous enterprise systems plug in without custom adapters

CALL LIFECYCLE

- User asks: "Epic 123 status?"
- LLM outputs JSON: `get_jira(id=123)`
- Engine calls real API → returns **Status: Open**
- LLM composes human-readable reply.

V2.0 Unlocks

LLM is no longer blind. It touches real pipelines — Jira, Git, CI/CD — through a unified protocol layer. → V3.0: What if the API times out? Add ReAct.

● **Tool Calling:** LLM → JSON command

● **Observation:** real result fed back

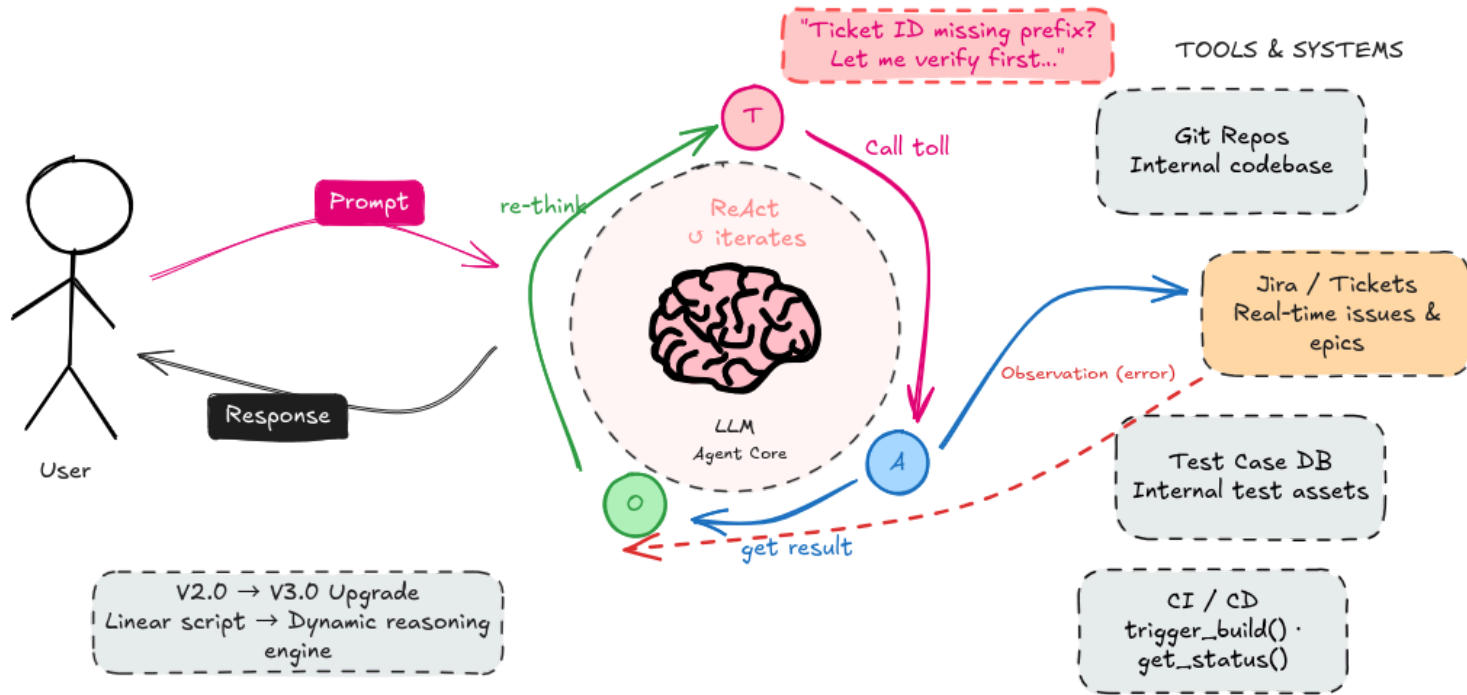
● **MCP:** one protocol, all systems

● **Next:** dynamic error recovery

Architecture V3.0: Implementing the ReAct Loop

Enabling Dynamic Planning and Error Self-Correction

V3.0



KEY OBSERVATIONS

CALL LIFECYCLE

- T Thought** - assess state, plan next step before acting
 - A Action** - call a specific tool with intent
 - O Observation** - feed raw result back, success or error
- Static IF-ELSE → Dynamic Loop
Each iteration re-evaluates context. No hard-coded failure paths.
 - Errors become inputs, not crashes
'404 Not Found' flows back as Observation — agent adapts its strategy.
 - CoT embedded in Thought
"Let's think step by step" runs before every action — fewer hallucinations.

V3.0 Self-Healing in Action
Ticket ID lookup fails → agent infers missing prefix → queries project list → retries with corrected ID.
No crash. No user-facing exception.

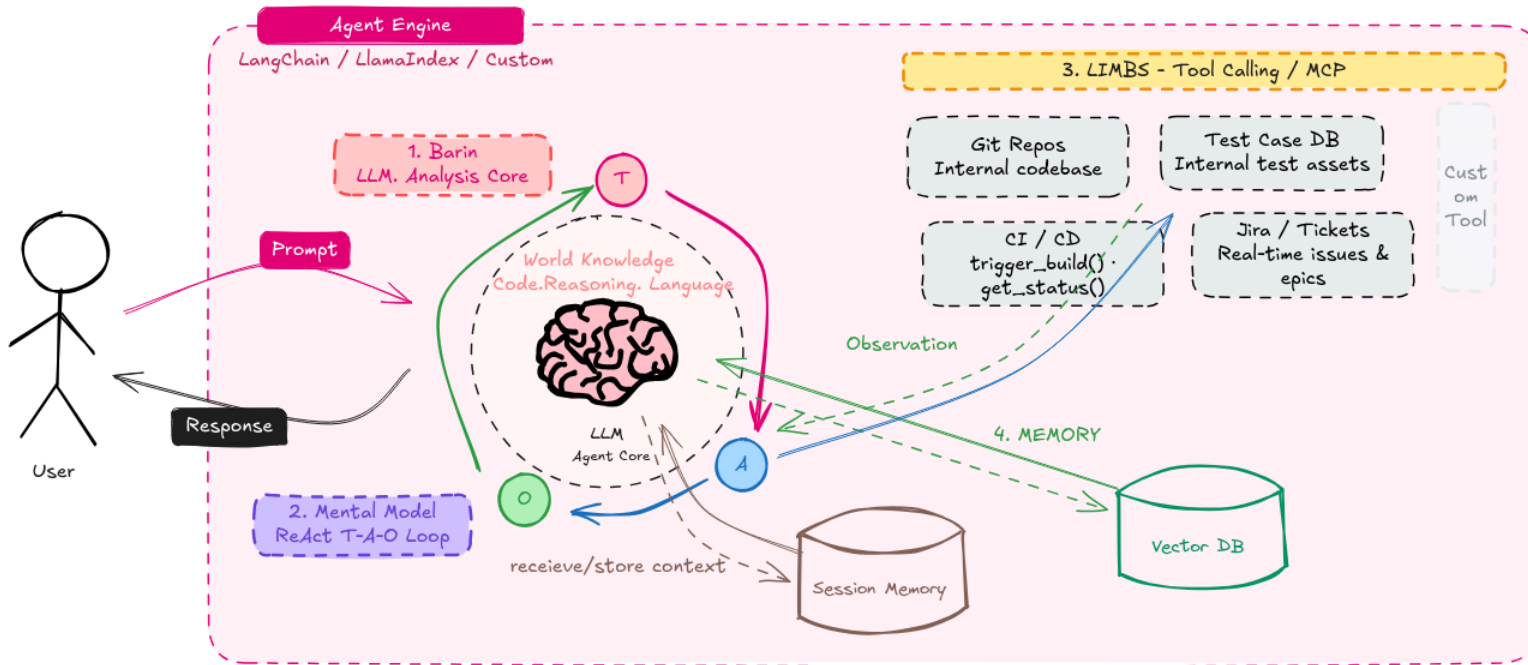
V4.0. Add persistent memory for long-horizon tasks

- Thought** - deliberate before acting
- Action** - targeted tool call
- Observation** - reality check, drives next cycle
- Failure** → feedback, not crash





Architecture V4.0: A Complete Single-Agent System

Consolidating Memory, Planning, and Execution Engines

V4.0



KEY PILLARS

-  **Brain**
LLM as analysis core. World knowledge + enterprise reasoning.
-  **Limbs**
Tool Calling + MCP. Unified interface for all enterprise systems.
-  **Mental Model**
ReAct T-A-O loop. Dynamic planning + self-healing on errors.
-  **Memory**
Session + Vector DB. Sprint context + enterprise knowledge.

Digital Worker
Plan · Execute · Recover · Remember
From fragile demo → production-grade agent
[Next: Live Demo](#) → SAFe sprint · real systems






- **Tool Calling:** LLM → JSON command
- **Observation:** real result fed back
- **MCP:** one protocol, all systems
- **Next:** dynamic error recovery

Business Context: Agile Workflows in SAFe

Addressing Bottlenecks in Epic Breakdown and Task Allocation

The Pain of SAFe

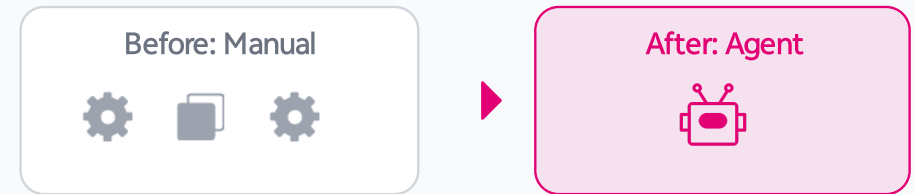
"SAFe ensures alignment but creates heavy manual work for engineers."

-  Start with Epic
-  Manual Decomposition
-  Manual Assignment
-  Cross-system Tickets
-  Copy-Pasting & Clicking

 Engineers as 'Cross-system Porters'

The Perfect Playground for AI Agents

"This context-heavy, multi-step coordination is where traditional scripts fail and agents excel."



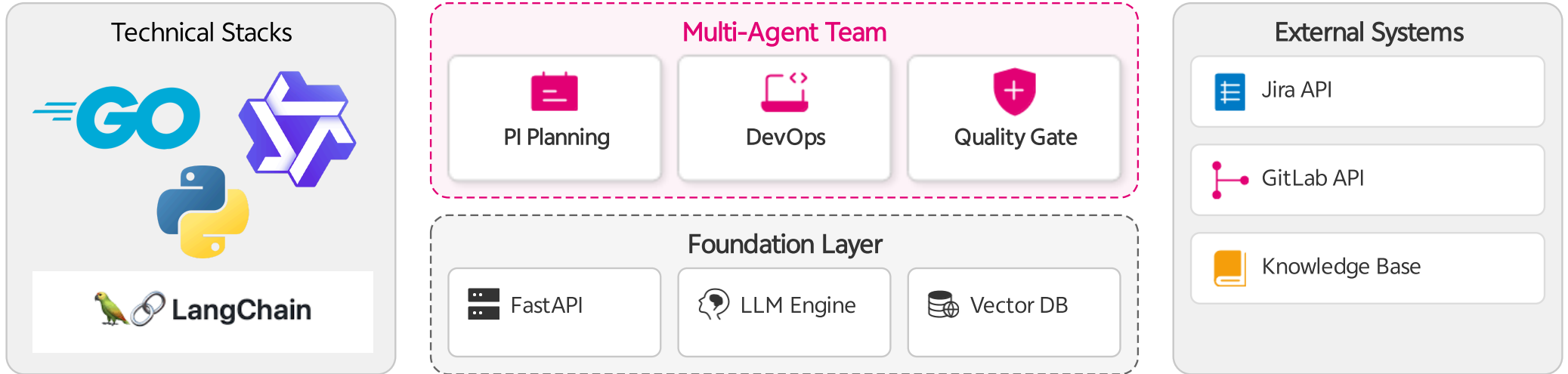
"From dozens of clicks to one natural language command."

Demo Note: This is a work in progress aiming to showcase potential automation.

"Let's see how AI Agents can turn repetitive coordination into strategic focus."

Mapping the Demo Architecture

Technical Stack and Enterprise System Integrations



Native ReAct & Self-Bootstrap

Custom-built control flow with ReAct, Tools, CoT, and Skill. Features 'self-bootstrap' - agent can write its own tools/skills based on needs.

Long & Short-Term Memory

Session memory + Vector DB for long-term knowledge. Includes 'timed session compression' for autonomous summarization and evolution.

Live Demonstration

Multi-Agent Collaboration in SAFe Scenarios

The screenshot displays the SAFE OS Multi-Agent dashboard. The interface is in Chinese and features a sidebar with navigation options: 总览控制台 (Overview Dashboard), 战略规划 (Strategic Planning), 开发运维 (Development & Operations), 质量门控 (Quality Gate), 项目与支撑 (Projects & Support), and 平台设置 (Platform Settings). The main content area is titled 总览控制台 (Overview Dashboard) and includes a search bar for 搜索需求、代码或 PR... (Search requirements, code, or PR...).

The dashboard provides a welcome message: 欢迎回来, SAFE OS 团队 (Welcome back, SAFE OS team) and a status indicator: AI 状态: 运行中 (随时可唤醒) (AI Status: Running (can be woken up at any time)).

Key metrics are displayed in three cards:

- 批量中需求 (Batched requirements): 24 (+3)
- 符合人 PRs (Compliant PRs): 12 (+2)
- 质量保障率 (Quality assurance rate): 92% (优秀) (Excellent)

The 核心工作流入口 (Core workflow entry) section contains three main workflow cards:

- 战略规划 (Strategic Planning):** 一站式需求管理与分析, 利用自然语言生成文档, AI 协助生成测试标准, 拆解任务并跟踪边界情况。
 - 智能需求分析与拆解
 - 上下文关联问题对话
- 开发运维 (Development & Operations):** 智能研发辅助流水线: 输入功能点描述即可生成规范代码框架和单元测试, 一键重运行测试链。
 - AI 代码框架生成
 - 自动化测试与建设
- 质量门控 (Quality Gate):** 项目质量的守护者, 提供可视化 Dashboard 概览, 智能审查 PR, 智能识别漏洞与缺陷。
 - PR 智能安全扫描
 - 对话式漏洞修复建议

At the bottom of each workflow card, there are buttons: 进入规划看板 (Enter planning dashboard), 开启开发流水线 (Start development pipeline), and 查看质量大盘 (View quality dashboard).

A large watermark "DEMO ONLY TECHNICAL SHARING 2026" is overlaid on the dashboard. A red button with a robot icon is located in the bottom right corner. The footer shows "项目管理员" (Project Manager) and "本地主机: 4000/devops" (Local host: 4000/devops).

[Click Here](#)

Live Demonstration

Multi-Agent Collaboration in SAFe Scenarios

The screenshot displays the SAFE OS Multi-Agent dashboard. The interface includes a sidebar with navigation options: 总览控制台 (Overview Dashboard), 战略规划 (Strategic Planning), 开发运维 (Development & Operations), 质量门禁 (Quality Gate), 设置与支持 (Settings & Support), and 平台设置 (Platform Settings). The main content area features a welcome message: "欢迎回来, SAFE OS 团队" (Welcome back, SAFE OS team) and a status indicator "AI 状态: 运行中 (随时可唤醒)" (AI Status: Running (Always on)). Three key performance indicators are shown: 任务中需求 (Tasks in Progress) at 24, 符合人天 (Compliance Days) at 12, and 质量保障率 (Quality Assurance Rate) at 92%. Below these are three core workflow entry points: 战略规划 (Strategic Planning), 开发运维 (Development & Operations), and 质量门禁 (Quality Gate). A large "DEMO ONLY" watermark is overlaid on the dashboard.

任务中需求: 24

符合人天: 12

质量保障率: 92%

核心工作流入口

- 战略规划**
一站式需求管理与分析, 利用自然语言生成文档, AI 协助生成测试用例、拆解任务和跟踪进度情况。
✓ 智能需求分析与拆解
✓ 上下文关联问题对话
- 开发运维**
智能开发辅助流水线, 输入功能点描述即可生成规范代码框架和单元测试, 一键回测测试验证。
✓ AI 代码生成生成
✓ 自动化测试与建议
- 质量门禁**
项目质量的守护者, 提供可视化 Dashboard 概览, 智能审查 PR, 拦截问题并与团队协作。
✓ PR 智能安全扫描
✓ 对话式漏洞修复建议

项目管理员

TECHSHARING DEMO ONLY

[Click Here](#)

Live Demonstration

Multi-Agent Collaboration in SAFe Scenarios

The screenshot displays the SAFE OS Multi-Agent dashboard in a web browser. The interface is organized into several sections:

- Header:** Includes the SAFE OS logo, the title "总览控制台" (Overview Control Panel), and a search bar.
- Left Sidebar:** Contains navigation options such as "总览控制台" (Overview Control Panel), "战略规划" (Strategic Planning), "开发运维" (Development & Operations), "质量门禁" (Quality Gate), "项目与支撑" (Project & Support), and "平台设置" (Platform Settings).
- Main Content Area:**
 - Welcome Message:** "欢迎回来, SAFE OS 团队" (Welcome back, SAFE OS team) with a note that AI Agents are ready for the day's tasks.
 - AI Status:** "AI 状态: 运行中 (随时可唤醒)" (AI Status: Running (可随时唤醒)).
 - Key Metrics:** Three summary cards showing "任务中需求" (Tasks in Progress) at 24, "符合人评审" (Compliance Review) at 12, and "质量保障率" (Quality Assurance Rate) at 92%.
 - Core Workflows (核心工作流入口):** Three main workflow cards:
 - 战略规划 (Strategic Planning):** Focuses on demand analysis and AI-generated recommendations. Includes features like "智能需求分析与拆解" (Intelligent demand analysis and decomposition) and "上下文关联问题对话" (Contextual question dialogue).
 - 开发运维 (Development & Operations):** Focuses on intelligent development pipelines and automated testing. Includes features like "AI 代码生成" (AI code generation) and "自动化测试与建议" (Automated testing and suggestions).
 - 质量门禁 (Quality Gate):** Focuses on project quality control and PR security. Includes features like "PR 智能安全扫描" (Intelligent PR security scanning) and "对话式漏洞修复建议" (Dialog-based vulnerability fix suggestions).
 - Bottom Section:** Includes "项目管理员" (Project Manager) and a "TECHSHARING DEMO ONLY" watermark.

[Click Here](#)

Execution Trace: Inside the Agent's Decision Process

Step-by-Step Analysis of the ReAct Loop in Action

SKILL ROUTING

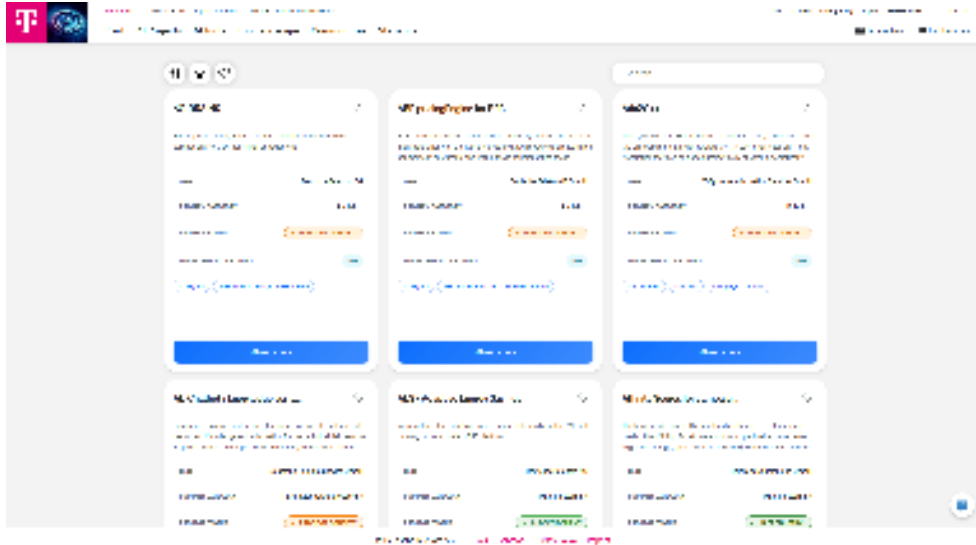
```
2026/03/16 12:40:40.859810 [INFO] [agent] trace_id=tr-189d389500471cfd-5f8fe5b35b8d6302 handle message stream chat_id=sess_8b20d2ac-87b6-4891-99e1-6f9912a9d306 user_id=user_a344d1b6-9a99-4b33-a87c-8cf6978446dc text_len=1169
2026/03/16 12:40:41.874570 [INFO] [llm.router] llm response success model=qwen3-0.6b output len=364
2026/03/16 12:40:41.874758 [INFO] [agent] trace_id=tr-189d389500471cfd-5f8fe5b35b8d6302 skill_router: need_skills=true requested=[SAFE 铁三角 PI 规划 文件系统查询专家 创建skill] matched=3 reason=需要规划 SAFE 铁三角 来制定 Epic 和 PI 规划，以实现云端 AI 分析和多端预警联动目标；同时需要文件系统查询来处理数据采集和权限检查；以及创建技能来完善相关功能。
2026/03/16 12:40:41.874775 [INFO] [agent] trace_id=tr-189d389500471cfd-5f8fe5b35b8d6302 skill_router selected 3 skills: [SAFE 铁三角 PI 规划 文件系统查询专家 创建skill]
2026/03/16 12:40:41.874876 [INFO] [agent] trace_id=tr-189d389500471cfd-5f8fe5b35b8d6302 unified react stream start
2026/03/16 12:40:41.875457 [INFO] [agent] trace_id=tr-189d389500471cfd-5f8fe5b35b8d6302 unified react stream start
2026/03/16 12:41:17.374147 [INFO] [llm] llm tool-call response success model=qwen3.5-plus content_len=3087 tool_calls=1 names=[publish_pi_plan] finish=tool_calls
2026/03/16 12:41:33.637488 [INFO] [llm] llm tool-call response success model=qwen3.5-plus content_len=4461 tool_calls=0 names=[] finish=stop
```

TOOL SELECTION AND EXECUTION

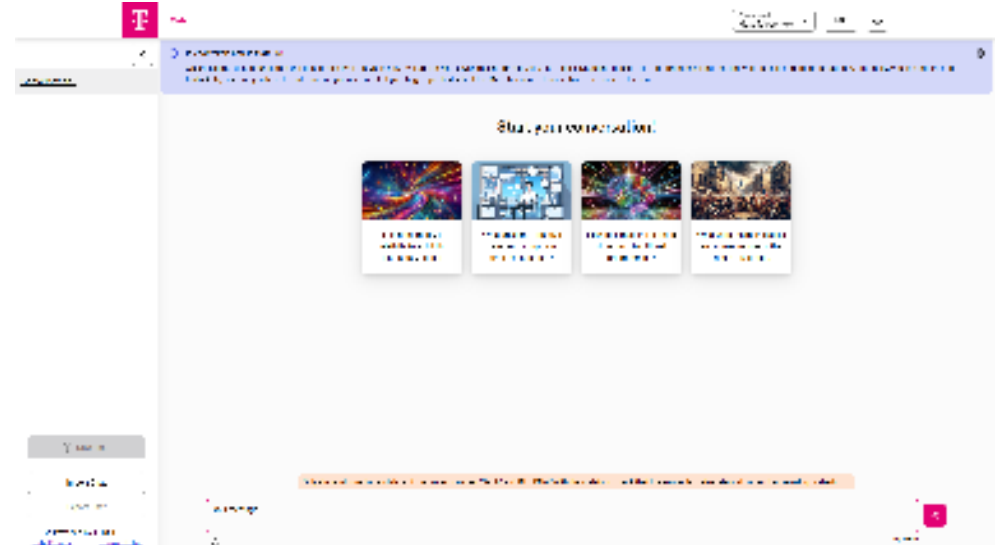
```
2026/03/16 12:42:33.721288 [INFO] [agent] trace_id=tr-189d38af3e141e0b-7f814b84c495e844 unified react stream start
2026/03/16 12:42:33.721984 [INFO] [agent] trace_id=tr-189d38af3e141e0b-7f814b84c495e844 unified react stream start
2026/03/16 12:42:59.570724 [INFO] [llm] llm tool-call response success model=qwen3.5-plus content_len=400 tool_calls=7 names=[create_gitea_ticket create_gitea_ticket create_gitea_ticket create_gitea_ticket create_gitea_ticket create_gitea_ticket create_gitea_ticket] finish=tool_calls
2026/03/16 12:43:11.189739 [INFO] [llm] llm tool-call response success model=qwen3.5-plus content_len=2032 tool_calls=0 names=[] finish=stop
2026/03/16 12:43:14.354277 [INFO] [agent] trace_id=tr-189d38af3e141e0b-7f814b84c495e844 stream message handled chat_id=sess_8b20d2ac-87b6-4891-99e1-6f9912a9d306 response_len=2032
2026/03/16 12:43:14.354334 [INFO] [transport.webui] webui stream chat completed session_id=sess_8b20d2ac-87b6-4891-99e1-6f9912a9d306 user_id=user_a344d1b6-9a99-4b33-a87c-8cf6978446dc reply_len=2032
2026/03/16 12:43:50.380085 [INFO] [agent] trace_id=tr-189d38c12091137e-72d12a3de5c878d8 handle message stream chat_id=sess_f55a889a-3946-49d1-9a7e-36d49cf8bd22 user_id=user_b75029cf-77e2-4a6a-9b6d-5defc9ff240c text_len=6
```

Production AI Applications

Moving from Proof of Concept to Real-World Value



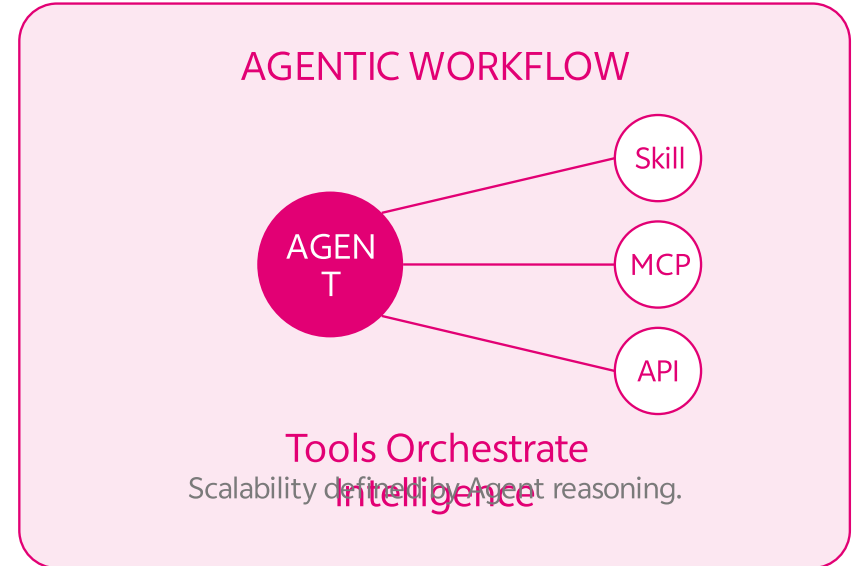
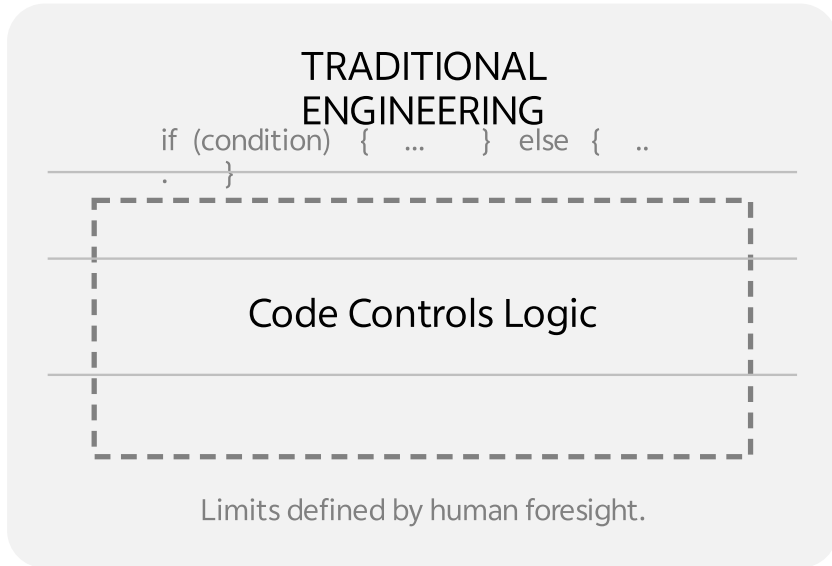
<https://aicompass.telekom.de>



<https://yourmate.telekom.com/>

Embrace the Agentic Workflow

Paradigm Shift: From Hard-coded Logic to Tool Orchestration



"Stop coding every branch.
Define the goal. Build the tools. Trust the agent."

Thank You!
Questions & Discussion Welcome
-- Bosch Tech Workshop 2026

THANK YOU!



Ding, Shuo

T-Systems P.R. China Ltd.
Digital Solutions Expert

Mobile +86 186 2776 9637
Shuo.Ding@t-systems.com