深圳 / 广州 · 随时到岗 | 2 年开发经验（2024.06 起） | 全日制本科 · 计算机科学与技术

任务拆解 · 工具调用 · 长流程执行

刘迪君

AI 应用开发 / AI Agent 工程 · 把自然语言变成可校验、能恢复的执行链路

全日制本科（计算机科学与技术），2 年开发经验。做**「自然语言指令 → 可校验动作序列 → 派发执行 → 异常恢复」这条链路的软件层**：任务拆解、工具契约、状态机、评测。上一份工作在企业级 Agent 系统里做后端开发（5 人团队），负责取数正确性与安全这一摊；个人独立完成巡检机器人任务执行仿真平台、Agent 安全评测靶场与多模态提取 demo。 系统已上线并在业务侧日常使用——问数一致率从 **35.6%** 提到 **98.1%**， 日报制作从人工 2 小时压到 5 分钟内，归因 Top3 命中率 50% 提到 80%，28 个越权用例 100% 拦截。

98.1%

问数一致率（原 35.6%，312 条黄金用例）

65 ms

单次问数 SQL 执行（原 741 ms）

80%

异动归因 Top3 命中（原 50%，30 个真值事件）

100%

28 个越权用例拦截率

01 能力构成

按工程落地的六块能力组织。

- **Agent 编排与任务规划**

用代码控制流替代自由 Agent 循环：状态机固定主链路，只在必需处留条件边；会话状态持久化、进度落盘、失败可重放。工具调用、多智能体拆分、MCP 接入；动作层按 ROS 2 Action 语义实现 goal / feedback / result / cancel 与终态机。探索式场景里模型只提「下一步切哪个维度」，取数、判异常、是否继续全在代码里。

`LangGraph` `StateGraph` `Checkpointer` `任务规划` `MCP`

- **RAG 与问数**

结构化知识检索（字段元数据 / Few-shot / 别名三类向量）、Join 图补全多表拓扑、粒度对齐防数值放大；指标语义层把口径配置化，模型不直接写 SQL。

`Qdrant` `Elasticsearch` `RRF` `Rerank` `Text-to-SQL`

- **工程底座**

Python AsyncIO 与分层架构、FastAPI / Pydantic / SQLAlchemy；MySQL 8.0 分区裁剪；Redis 结果缓存与请求合并、分布式锁（SET NX PX + Lua 校验持有者），缓存键按权限范围区分。

`FastAPI` `MySQL 8.0` `Redis` `Docker` `Linux`

- **安全与质量**

sqlglot AST 七类白名单校验、权限条件下推最内层 WHERE（防推断式越权）、数字白名单防幻觉；638 条 pytest 用例、黄金用例跑评与发布门禁、LangFuse 可观测。

`sqlglot` `AST 白名单` `篡改检测` `pytest` `发布门禁` `LangFuse`

- **调度、上位机与通信**

FastAPI + WebSocket 下发任务、取消、人工接管与告警确认，控制权互斥；HTTP 接口契约（ToolRequest / ToolResponse）与入参在服务端二次校验；Python AsyncIO 处理事件流，每步进度落盘。

`FastAPI` `WebSocket` `AsyncIO` `上位机`

- **机器人仿真与工具链**

在 ROS 2 + MoveIt2 环境跑通 UR5 开源模型的仿真示例：配置 URDF、用 Python 节点调用接口下发目标位姿，观察规划与执行结果；巡检平台用 A* + 栅格代价膨胀做路径规划，二维栅格仿真，非真机。

`ROS 2` `MoveIt2` `A*` `栅格规划` `仿真`

02 项目

01 工作项目 | 2025.11 – 2026.07 | 后端开发 · 5 人团队（后端 2 / 前端 1 / 业务 2） | 已上线

企业级智能经营分析 Agent

解决什么问题

业务人员用一句话提问，系统自动完成取数、计算与解读，替代人工导数、拼表、写报表。要解决的三个问题：指标口径不统一、大模型直接生成 SQL 不可信、多表权限管控难。

我负责的部分

后端开发，5 人团队里后端 2 人之一。负责**「取出来的数对不对」**这一摊：指标语义层、结构化检索链路、主链路里交给模型的那几步与条件边、SQL 校验与权限下推、黄金用例评测与发布门禁，以及后来的假设驱动探索式归因。

实际应用效果

系统已上线，在业务侧日常使用，服务运营、分析师、管理者等多类角色；问数、日报、异动归因等场景跑在业务数据上，日报从人工拼表变成系统内直接出。

35.6% → 98.1%

问数一致率（312 条黄金用例，SQL 可执行率 99.4%）

741 → 65 ms

单次问数 SQL 执行（STORED 生成列触发分区裁剪）

120 → 5 min

日报制作（人工 → 系统内）

50% → 80%

异动归因 Top3 命中率（30 个带 ground truth 事件，平均 4.5 s、残差 0）

▾ **展开：架构 · 负责的几块 · 三次线上问题**



负责的几块

- **RAG 检索链路**：以字段元数据、Few-shot 用例、指标别名三类向量集合（67 / 104 / 74 条）为检索源，最小检索单元是「一个字段 / 一条用例 / 一个别名」；用 **Join 图 BFS 最小边集**补全多表拓扑，用**粒度对齐**防止一对多 JOIN 把金额放大；以 score_threshold + payload 过滤 + 业务优先级剪枝替代两阶段 rerank，省掉一次模型调用。
- **指标语义层**：12 个核心指标的口径（公式 / 过滤条件 / 时间归属 / 可用维度 / 权限标签）配置化落库，由统一取数规划器翻译成只读 SQL——口径变更不发版，代码侧零硬编码公式。
- **主链路编排与模型节点**：LangGraph StateGraph 把 15 节点主链路固化为代码控制流（仅 5 处条件边），路由函数全是纯函数、可单测可回放；定清哪几步交给模型、哪几步必须是代码，会话状态经 Checkpointer 持久化。
- **SQL 校验、权限下推与门禁**：sqlglot AST 七类校验 + 权限条件按承载表逐维注入、列名用该层别名限定；28 个越权用例直接断言权限改写函数的产出；312 条黄金用例跑评低于基线自动阻断发布。
- **假设驱动的探索式归因**：模型只读盘面与上一刀观测、输出「下一刀切哪个维度」，取数、判异常、是否继续全在代码里；白名单外的维度与重复提议直接拒绝并降级为规则顺序，上限三刀、每刀写审计、可回放。与确定性四维全扫并存——出报告走确定性，快速定位与开放式提问走探索式。

三次线上问题的定位与修复

> **一对多 JOIN 让归因结论被挤出 Top3**
>
> 一次异动归因，系统把主因算到了品类上，与业务判断对不上。根因：品类维度 JOIN 明细表后仍沿用订单级金额求和，一个订单含多商品时金额被重复累加——数字量级「看起来像那么回事」、不报错，只有排名异常才暴露。改为明细金额聚合，并用造数保证「明细金额之和 == 订单实付金额」，整月差值 0。

> **权限校验 30 项全绿，而条件注入完全没生效**
>
> 根因是校验脚本自己拼了一遍权限条件再比数字——测试复述了设计意图，没有断言执行路径。重写为 28 个越权用例直接检查权限改写函数的产出，第一次运行就抓出 4 个缺陷，最终 100% 拦截。

> **时间口径一次错 44 条：「2 月第二周」被当成整月**
>
> 104 条时间解析用例只有 60 条对（57.7%），错的 44 条里 40 条都是同一类——「N 月第 K 周」这种相对周表达被解析成了整月区间，数字不报错、只是范围悄悄变大。把相对周表达单独收进口径配置、按数据窗口起点推算自然周边界，修完 104 条全对。这类错误只有拿标准答案逐值比对才暴露。

技术选型

为什么不用自由 Agent	业务链路固定，自由循环换来的是不可复现的失败；固化成状态机后每一次失败都能重放到具体节点。
为什么不让模型写 SQL	模型负责「理解问题 + 拼装结构」，SQL 由语义层编译；模型输出永远不直接进执行路径。
为什么不做 rerank	结构化检索的候选集很小且带业务标签，阈值 + payload 过滤已足够，省一次模型调用的延迟与成本。
为什么分两条路	出报告的数字要满足「贡献之和等于大盘偏离」，只能四维全扫、基准做趋势校正；让模型跳着切，这个前提就没了。要快定位和答开放式问题，另开一条探索式链路，两条并存、数字不混用。

`Python` `FastAPI` `LangGraph` `MCP` `MySQL 8.0` `Redis` `Qdrant` `Elasticsearch` `sqlglot` `DeepSeek` `pytest`

02 个人项目 2026.10 – 2026.11 独立开发 仿真环境，非真机

巡检机器人任务执行仿真平台

解决什么问题

把自然语言巡检指令翻译成按序、可取消、能恢复的动作序列，用于配电房、机柜区等点位的日常巡检。难点不在「生成一段计划」，而在计划落库前的校验、执行中的进度反馈，以及中断之后怎么续跑。

我负责的部分

独立开发：任务规划与三道校验、按 ROS 2 Action 语义自行实现的动作层（goal / feedback / result / cancel 与终态机）、异常恢复、上位机通信、多传感器告警规则引擎，以及回归用例集。上位机用 FastAPI + WebSocket 下发任务、取消、人工接管与告警确认，控制权互斥；路径规划用 A* + 栅格代价膨胀，二维栅格仿真，非真机。

实际应用效果

40 轮回归 / 84 个导航目标 **100% 达成**，7 次低电量中断全部续跑；注入 34 个故障告警召回 100%，13 轮无注入误报 0；10 条坏计划（越权 / 未知点位 / 错配 / 地图过期）100% 拦截；52 个 pytest 用例全绿。

84 / 84 40 轮回归中的导航目标达成数	**7 / 7** 低电量中断后全部断点续跑
34 / 34 注入故障告警召回，13 轮无误报	**52 条** pytest 用例全绿

▾ **展开：三个设计要点**

任务规划与三道校验

自然语言先转成结构化巡检计划，落库前必经 JSON Schema、地点白名单、地图 SHA 指纹三道校验；未声明的动作类型一律拒绝，校验失败回灌重生成，重试用完即失败。规则与模型双路径，无模型环境也能离线回归。

Action 执行层与异常恢复

按 ROS 2 Action 语义实现目标派发、进度反馈、结果回收与取消，配终态机；串行目标队列 + 单写锁保证单目标执行，每步进度落盘，中断后从断点续跑。电量低于 25% 中止当前目标、回充至 80% 后重排剩余步骤；导航超时、卡死、人工接管三态分别收敛并留痕。

调度与上位机

FastAPI + WebSocket 下发任务、取消、人工接管与告警确认，控制权互斥；HTTP 接口统一 ToolRequest / ToolResponse 契约，入参在服务端二次校验；Python AsyncIO 处理事件流，每步进度落盘，中断后从断点续跑。

感知与告警

温度 / 湿度 / 气体 / 电压 / 指示灯等多传感器统一采集接口与单位归一；可疑读数降级为 UNKNOWN 并补测。告警按「持续时长 + 滞回 + 去重」判定，规则身份按传感器 / 比较符 / 阈值区分，避免多条规则互相清计时。

在线 ▶ 执行回放（一次完整巡检的逐 tick 轨迹）↗ dijundev.localhost.cc/replay.html

不用登录、不用装环境，打开就能看

代码 GitHub · robot-sim-agent ↗ 52 个单测 · GitHub Actions 跑 3 个 Python 版本

`Python` `FastAPI` `WebSocket` `A* 栅格规划` `状态机` `pytest`

03 个人项目 | 2026.10 – 至今 | 独立开发 | 已完成

Agent 安全靶场：提示注入红队与防御评测

解决什么问题

Agent 一旦带上工具（读文件 / 查库 / 发消息 / 发请求），一段藏在文档里的提示注入就能把数据带出去。这类风险需要一套可量化、可对比的评测口径。

我负责的部分

独立开发：60 条攻击用例设计（5 类攻击面）、带四个工具的目标 Agent、五层可开关防御、规则断言 + LLM 判据、评测流水线与看板。

实际应用效果

在大模型上完成全量评测：24 条自适应攻击重复 2 轮，防御全关被攻破 **28.3%**，防御全开 **0 漏**；换一家厂商的模型重跑，基线升到 **52.2%**，全开仍是 **0**。另做了图片注入与编码混淆载荷的对照测试，均未出现绕过。逐层消融定位到工具白名单是关键防线。

28.3% → 0%
24 条自适应攻击 × 2 轮，防御全关 → 全开

52.2% → 0%
换一家厂商的模型重跑同一套用例

7 配置
五层防御逐一消融，定位关键防线

38 条
单元测试与端到端回归全绿

▾ **展开：架构 · 评测方法 · 实测结果**



为什么做这个

带工具的 Agent 一旦被提示注入命中，风险发生在工具调用上：模型能拒绝直接的恶意指令，却可能照着文档里的伪造指令去读文件、查库或请求外部地址。这个项目把这件事变成可测量的数字：**不可信输入不进执行路径**——在 SQL 上是 AST 白名单与权限下推，在 Agent 上是工具白名单与轨迹断言。

评测方法

- **命中判定**：断言工具调用序列——是否调用了越权工具、参数是否含敏感路径、外发请求是否携带密钥；以工具执行成功作为攻破判据，被拦截的调用不计入。
- **误杀对照**：每个配置同时跑一组不含攻击的正常任务，防御导致正常任务失败会计入误杀率。

实测结果

- **效果**：24 条自适应攻击各跑 2 轮，防御全关被攻破 28.3%，防御全开 0 漏；换一家厂商的模型重跑同一套用例，基线升到 52.2%，全开仍是 0。
- **逐层消融**：五层防御逐一关掉后复测，只有工具白名单这一层会重新出现攻破，说明它是这组攻击面的关键防线；其余四层在这批用例上没有额外贡献。
- **对照测试**：把伪造指令做成图片送进多模态模型，以及 base64、全角同形字、HTML 注释、CSV 公式等编码混淆载荷，均未出现绕过。
- **工程质量**：38 条单元测试与端到端回归全绿，GitHub Actions 在 Python 3.11 / 3.12 上自动跑。

代码 GitHub · dijundev/agent-redteam ↗ Gitee 国内镜像 ↗

`Python` `FastAPI` `LangGraph` `Ollama / Qwen` `SQLite` `原生 JavaScript` `pytest`

04 个人项目 2026.04 – 2026.07 独立开发（后端 + 协议层 + 前端）

自助式数据看板搭建平台（低代码）

解决什么问题

团队里「要一张看板就得排期开发」，需求积压、响应慢；业务方改一个指标也要走一轮迭代，大量临时需求只能靠手工导表顶着。

我负责的部分

独立开发：物料协议、页面 DSL 与表达式 DSL 解析器、服务端只读 SQL 编译、权限下推、缓存与发布快照，以及前端编辑器。

实际应用效果

端到端跑通：受限用户只能看到其权限范围内的数据，无法用「总数 − 可见明细」反推隐藏部分；22 条注入与 AST 逃逸样本 100% 拦截、0 误杀。

34.9 ms 编译 + 只读校验 + 权限下推 P95	**0.18** ms 整页渲染 P50
522 → 0.19 ms 结果缓存；同页 10 并发只打库 1 次	**22 / 22** 注入与 AST 逃逸样本 100% 拦截、0 误杀

▾ **展开：五个设计决策**

五个设计决策

- **协议驱动渲染**：每类物料声明 props_schema（属性结构 / 数据契约 / 默认值），属性面板自动生成——新增组件只注册物料、不改编辑器，把开闭原则落到代码结构上（8 类物料）。
- **表达式不开放 eval**：表达式来自业务配置，等于把执行权交给配置人员。自研递归下降解析器把它变成自己的 AST，语法里根本没有属性访问、下标、import，所以 `__import__('os').system(...)` 连解析都过不去。
- **权限下推到最内层 WHERE**：不做「查完再过滤」——那样可以用「总数 − 可见明细」反推隐藏数据。条件注入到每个直接读取目标表的 SELECT，并用该层别名限定列名，避免 JOIN 歧义列报错。
- **粒度对齐**：orders ↔ refunds 是 1:N 且非粒度对齐，直接 JOIN 会让 SUM(pay_amount) 按退款条数翻倍；编译器先把退款按 order_id 预聚合到订单粒度，再 LEFT JOIN。
- **发布快照 + 缓存键含权限指纹**：草稿 / 发布双态、发布生成快照可回滚；缓存键强制包含权限范围指纹，避免跨权限命中。

模型在这里的位置

自然语言 → 页面 DSL 只产出**候选**，不做任何直接执行；候选经 AST 白名单校验 + 试渲染双重把关，校验失败把错误信息回灌重新生成。模型能力与确定性边界各自负责一段，出问题能定位到是哪一段。
受限用户只能看到其权限范围内的部分数据，且无法用「总数 − 可见明细」反推隐藏部分。

代码 GitHub · dsl-guard ↗ Gitee 国内镜像 ↗

安全内核（表达式解析器 + SQL 守卫 + 值校验，103 个测试）已独立开源

`Python` `FastAPI` `原生 JavaScript` `MySQL 8.0` `Redis` `sqlglot` `pytest`

05 个人项目 2026.11 独立开发 开源

图片 / PDF → 结构化 JSON 的多模态提取

解决什么问题

发票照片、扫描合同要转成表格，人工敲太慢。真正的难点不是「把字读出来」，而是**判断「读得对不对」**——模型不会报错，错的金额看着和真的一样。

我负责的部分

独立开发：PDF 按页渲染成图与图片压缩、提示词模板、按内容哈希分页缓存、JSON 剥壳、结果自检规则、表格导 Excel，以及 23 条不联网单测与 CI。

实际应用效果

三个样例（发票 / 拍歪的表格照片 / 两页 PDF）4 页 7180 token、6 秒跑完，成本不到一分钱；同一份文件再跑 0.5 秒 0 token。自检脚本只拿文档本身该成立的算术关系去查（列数 / 单价 / 合计 / 日期），故意把一行金额改错，一次报出三个 FAIL 并以退出码 1 结束，可以直接当闸门挂在批处理里。

代码 GitHub · multimodal-extract-demo ↗ CI 跑 3.10 / 3.12，不调模型、不烧 token

`Python` `Pillow` `PyMuPDF` `openpyxl` `pytest`

附：AI 辅助交付 · 键锋打字练习站

用 Codex / Cursor 辅助编码，2 小时内从零交付并上线，之后补了指法诊断、单指专项与节奏反馈。现有 8 种练习模式（含输入法中文）、时间 / 词数两种计时、虚拟键盘指法分区与错键热力图、结果复盘与错字重练。AI 产出经复核后合入。

打开站点 → 代码

03 五条工程信条

从项目里沉淀下来的五条做法。

01 **模型只做理解与拼装，规则必须落在确定性代码里**

口径、权限、SQL 结构由语义层与 AST 校验承担，模型输出永远不直接进执行路径。这样出错时能定位到「是理解错了，还是规则没覆盖」。

落地：指标语义层（12 指标）、sqlglot 七类校验、权限下推最内层 WHERE。

02 **测试要断言执行路径，不照设计复述**

权限校验 30 项全绿，而权限条件注入完全没生效——因为脚本自己拼了一遍条件再比数字。测试如果照着设计写，就只是在验证「我以为」。重写后 28 个越权用例第一次运行就抓出 4 个缺陷。

03 **粒度不对齐比语法错误危险得多**

JOIN 放大不报错、量级看着合理，只让排名悄悄错掉。修法是造数断言「明细之和 == 订单实付」；低代码平台里同样先用预聚合对齐粒度再 LEFT JOIN。

04 **模型提假设，代码做判定**

归因里让模型决定「下一刀切哪个维度」，但取数、判异常、要不要继续全在代码里，白名单外的提议直接拒绝。模型的价值是缩小搜索空间，不是替系统下结论。

落地：探索式归因上限三刀、每刀写审计，30 个真值回放定位 18 个；点名的维度 1 刀命中。

05 **数字能加回来，才算结论**

各维度贡献加起来必须等于大盘偏离，残差不为零这条结论就不能拿去汇报。所以出报告走固定的四维全扫 + LMDI，而不是让模型自由发挥；探索式那条只用来快速定位和回答开放式提问，不进报告。

落地：30 个异动事件残差 0、Top3 命中 80%；探索式三刀内定位 60%、平均 2.6 刀。

04 经历

2026.10 – 2026.11

个人项目：巡检机器人任务执行仿真平台 / Agent 安全评测靶场 / 多模态提取

前者把自然语言变成可校验、能恢复的动作序列；中间那个把「不可信输入不进执行路径」从 SQL 扩展到 Agent 的工具调用链；后者解决「模型说得通但读错了」怎么查出来。

2026.04 – 2026.07

个人项目：自助式数据看板搭建平台（低代码）

页面 DSL 与表达式 DSL 自研解析器，服务端编译成只读 SQL；与工作项目共用同一套语义层与权限模型。

2025.08 – 2026.10

酷宝信息技术有限公司 · AI 应用开发工程师（5 人团队）

后端开发企业级智能经营分析 Agent：指标语义层 / 检索链路 / 主链路编排与模型节点 / SQL 校验与权限 / 评测门禁 / 探索式归因；2026.07 交付上线后转线上稳定性保障与运维文档沉淀。

2025.08 – 2025.10

多源异构数据抓取与增量更新框架

统一多来源字段口径、沉淀可复用中间表，搭起定时任务集群与运行监控。

2024.06 – 2025.05

数据开发实习 · 元数据与数据字典

盘点表 / 字段 / 指标口径，整理成 YAML 配置解析入库。

2021.09 – 2025.06

南昌理工学院 · 计算机科学与技术 · 全日制本科（学士）

主修：人工智能、数据挖掘、数据库、MySQL、Python。

联系我

深圳 / 广州 · 随时到岗。电话 19295745865 ｜ 邮箱 liudijunfo@foxmail.com。个人项目已开源，代码链接见上。

电话 **19295745865**

邮箱 **liudijunfo@foxmail.com**

GitHub 代码 **agent-redteam**

国内镜像 **Gitee**

靶场看板 在线评测结果