FIELD GUIDE / 管理咨询 × AI落地

从华为 LTC 到 Ontology 与 Agent

企业 AI 转型的诊断、设计与实施

> 先看清一笔生意怎样运行，
> 再决定让 AI 改变哪一段工作。

核心讲解 · 官方图示 · 完整口播 · 实践工作表

小红书

心晴九天

小红书号：howru999

我的 Agent

扫描二维码进入

Agent 入口

阅读指南

怎么使用这份资料

从企业诊断出发，沿着同一笔生意连接管理与工程。

01 / 核心讲解

25张主题卡，覆盖经营诊断、价值流、企业架构、目标运营模式、Ontology、受控执行和试点验收。

02 / 完整口播

保留完整讲述正文，方便复习与对照演示。

03 / 实践工作表

用诊断卡、动作定义卡和验收清单，把内容带回自己的项目。

04 / 延伸阅读

列出官方图示来源与陈果文章的参考观点，区分原始材料和本资料的综合分析。

使用边界

LTC是从线索到回款的业务流程体系，可从价值流视角分析；它并不等同于管理咨询。文中的开票情境是教学例子，具体条件应以实际合同为准。

华为图示来自2013年公开白皮书，属于历史参考。Palantir截图是厂商示例，不代表本人的客户上线成果。本资料未包含客户身份、录音、内部经营数据或案例细节。

核心讲解 01 / 为什么现在

把人的判断
交给 Agent 之前

隐含经验，需要变成明确的执行依据

依据什么判断

从“员工心里有数”到有据可查

谁有权决定

授权不会因为模型更聪明而消失

怎样才算完成

看业务状态，不只看一段回复

依据什么判断。 谁有权决定。 怎样才算完成。

原创分析 · 教学示意

核心讲解 02 / 经营诊断

项目名称 ≠ 问题根因

先定位业务卡点，再决定做什么 AI

回款慢

第一反应：做一个催收助手

客户还没验收

多发提醒，能解决吗？

比如回款慢，第一反应可能是做一个催收助手。 但客户一直没有验收，多发提醒作用可能有限。

原创分析 · 教学示意

核心讲解 03 / 经营诊断

把回款周期拆开看

先交付、验收、再开票付款的项目示例

交付 → 验收

查履约是否满足要求、证据是否齐全

验收 → 开票

查资料准备和内部处理

约定付款日 → 实际收款

区分约定账期、逾期与争议

从交付到验收，时间花在哪里？ 验收以后，开票有没有耽搁？ 合同约定何时付款，是否逾期？不同付款安排按合同分析。

教学诊断示意 · 非通用付款顺序或 DSO 公式

核心讲解 04 / 投入优先级

容易自动化
≠ 值得先投入

价值驱动分析 · 从经营目标找到可干预因素

影响范围

多少业务受到影响

损失大小

周期、成本与机会损失

改善成本

实施与长期维护投入

沿着整笔生意追原因

不要只盯最后出问题的部门

比较影响范围。 比较损失。 比较改善成本，排出投入顺序。

核心讲解 05 / LTC价值流

一笔生意
从需求到兑现

华为公开架构：线索管理 / 机会点管理 / 合同履行管理

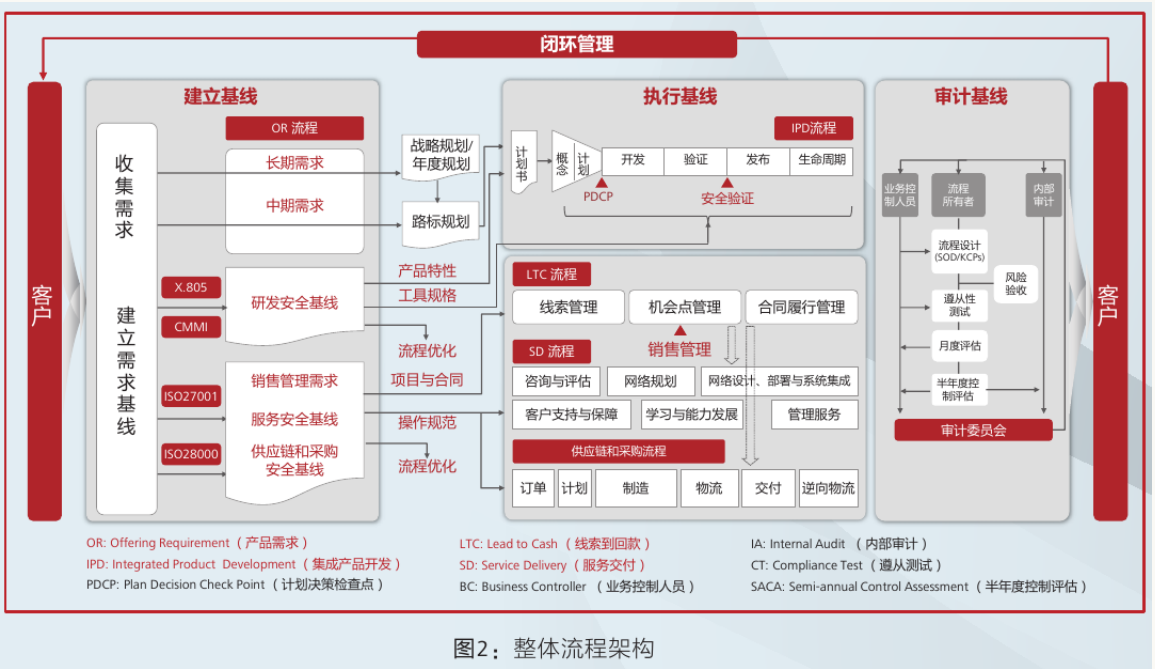


图2：整体流程架构

线索管理

机会点管理

合同履行

判断需求是否值得跟进。 形成方案与商业条件。 把约定的事做完，完成相应收款。

华为《网络安全透视》2013 · 图2；说明为作者展开

核心讲解 06 / 视角与边界

业务看全程
系统有分工

价值流与具体流程，是不同观察视角

价值流

客户价值 → 商业结果

具体流程

合同评审 / 资源申请 / 开票申请

CRM

机会

合同系统

条款 / 评审

项目系统

资源 / 交付

ERP

财务记录

流程规定谁在什么条件下完成工作。 CRM与合同系统各自分工。 项目与ERP继续承担相应职责。

原创分析 · 教学示意

核心讲解 07 / 企业架构

四种架构
对齐一件事

业务准备怎样运行，系统就要怎样支撑

业务架构

业务怎么运行

DA / IA

数据架构

信息如何定义

应用架构

系统各自负责什么

TA

技术架构

能力如何支撑

先利用成熟能力

真正缺的部分，再补

业务与数据要对得上。 应用分工与技术支撑也要对齐。 不必每次从零画宏大蓝图。

原创分析 · 教学示意

核心讲解 08 / 目标运营模式

谁来组织这笔生意？

Target Operating Model · 把工作方式设计清楚

流程与分工

谁做事，谁推动跨部门协作

决策与授权

谁能批准取舍，超出权限交给谁

绩效与结果

共同关注履约，也保留岗位可控指标

流程负责人不能只是催进度。 超出权限，需要裁决路径。 考核要配合，避免各部门达标、项目却没做好。

核心讲解 09 / 管理设计

把规则
写到能执行

管理上的模糊，会变成 Agent 的反复确认

进入 / 退出条件

什么证据允许进入下一步

授权与例外

允许自动处理到哪里

负责人

遇到分歧，谁作决定

明确进入和退出条件。 明确授权和例外。 否则Agent仍然要不停找人确认。

核心讲解 10 / 业务语义

同一笔生意
不同系统身份

不能靠名称相似，就认定是一回事

CRM

机会编号

合同系统

合同号

交付系统

项目号

先明确对应关系与记录来源

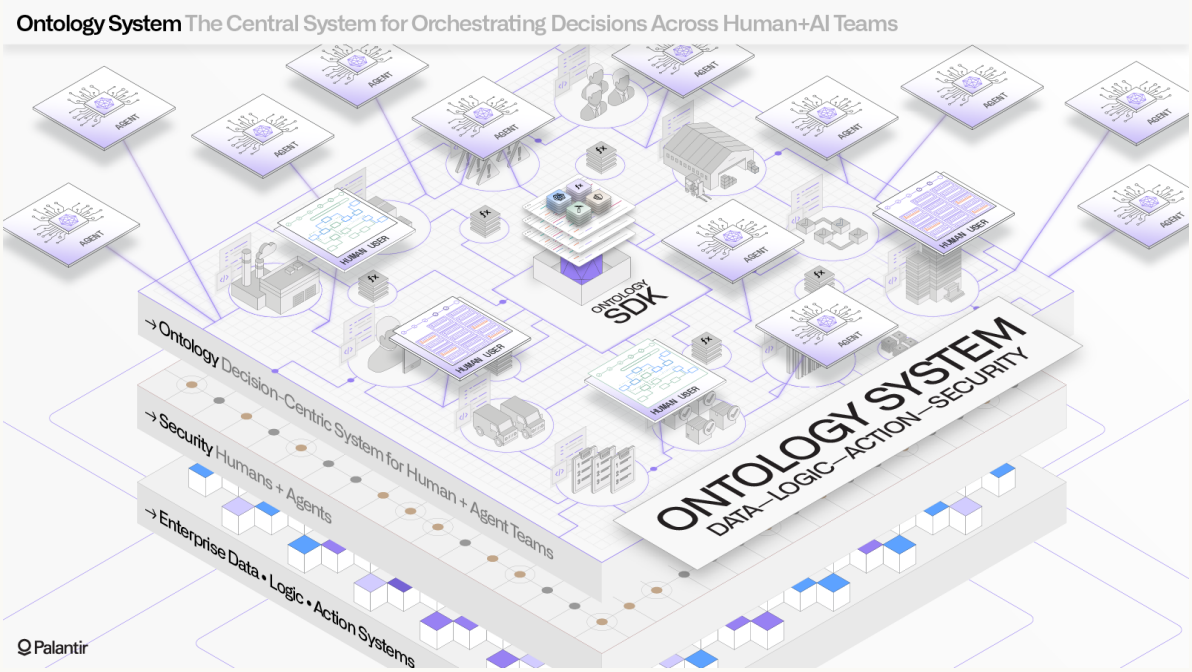
CRM有机会编号。 合同系统有合同号。 交付又有项目号，Agent不能靠猜测关联。

原创分析 · 教学示意

核心讲解 11 / Ontology

让业务含义有共同定义

Ontology · 对象、关系与业务含义



业务对象

对象关系

共同语义

再连接规则、权限与动作

用明确对象与关系组织共同理解。 面向执行时，需要连接动作与约束。

Palantir · Ontology overview · 官方概念图

核心讲解 12 / 概念分工

材料、入口与业务含义

能力可以集成在同一平台，不是互斥产品

知识库

帮助找到相关材料

API

提供查询与操作入口

Ontology

说明对象关系、状态与含义

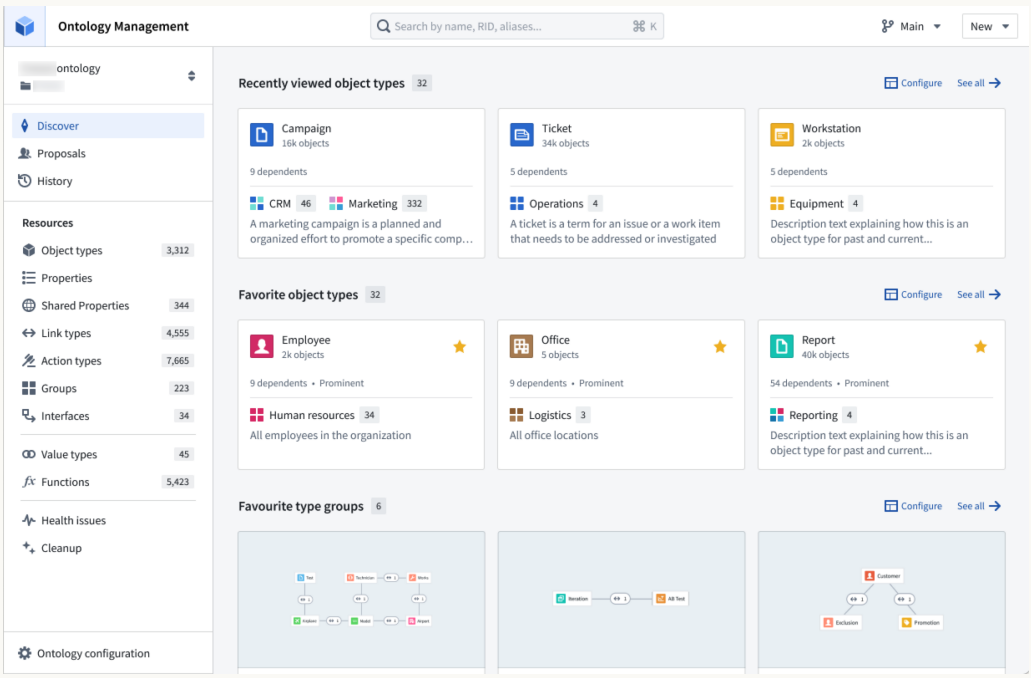
找到材料。 提供操作入口。 共同理解“是哪份合同，完成到哪一步”。

原创分析 · 教学示意

核心讲解 13 / 官方产品画面

共同定义
怎样被管理

示例：Ontology Manager



对象类型 · 关系 · 可用动作

多个应用复用同一套业务定义

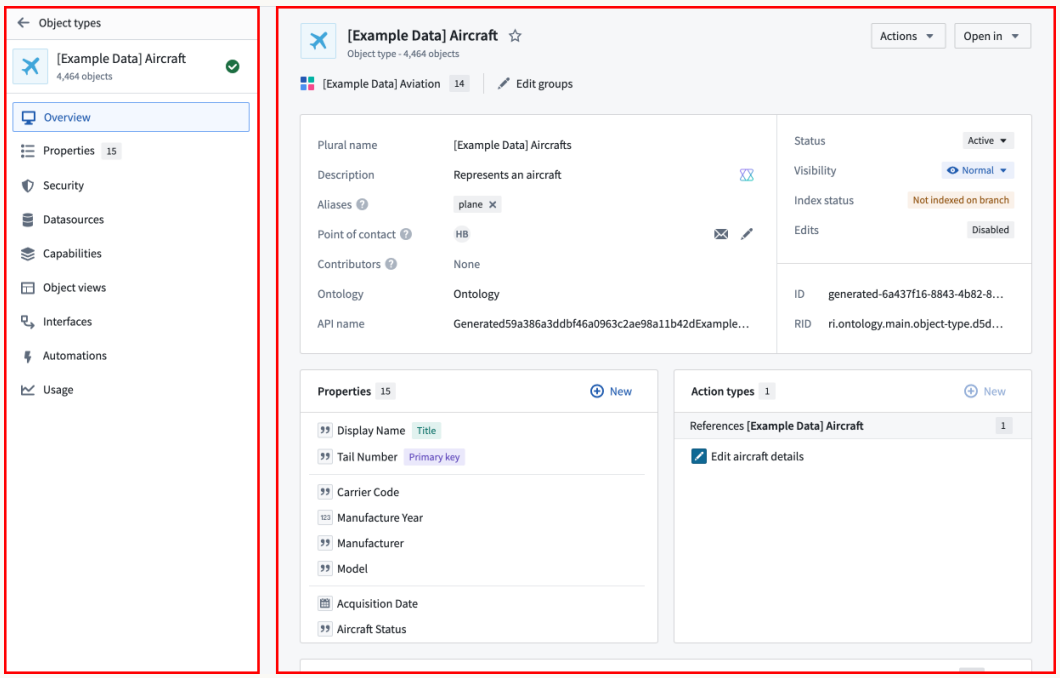
看官方管理界面：对象类型与关系集中组织。 这是产品实现参考，不是本项目已上线成果。

Palantir · Ontology Manager 文档 · 官方产品截图

核心讲解 14 / 对象与属性

不是同名
就是同一对象

官方界面中的类型与属性配置



名称、属性、关系，各有定义

简单场景先用现有模型

不必先建一套大平台

对象类型和属性需要明确。 简单场景的现有模型可能已经够用。

Palantir · Ontology Manager · Example Data 截图

核心讲解 15 / 状态判断

“完成”
究竟完成了什么？

同一个词，不能替代业务判断

交付完成

工作是否完成

客户验收

客户是否按约定确认

开票 / 到账

各自有条件与记录来源

交付结束。 客户已验收。 允许开票和款项到账，不是同一种状态。

原创分析 · 教学示意

核心讲解 16 / 从理解到行动

开票申请
需要什么依据？

教学示例 · 按实际合同付款安排判断

对应哪份合同、哪个节点

对象与关系

约定条件是否满足

引用相应履约或付款条件证据

发起获准的申请动作

有预付款时，不能机械要求先验收

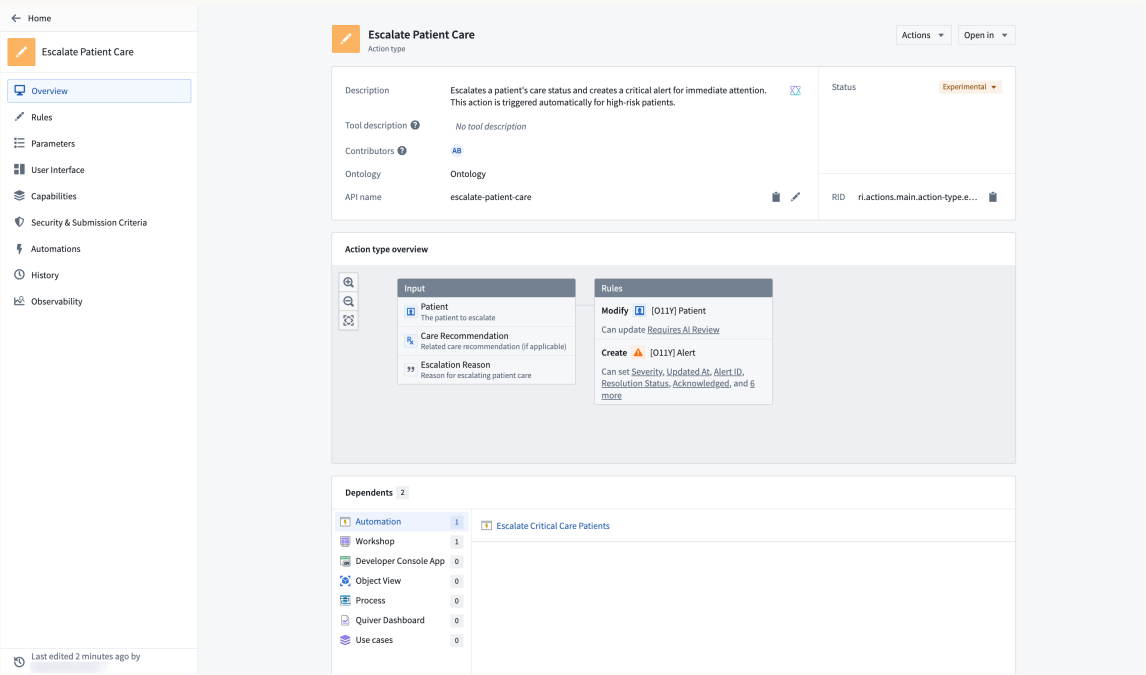
找到合同与付款节点。 检查约定条件，引用证据。 再发起申请；预付款不能套用先验收规则。

原创分析 · 教学示意

核心讲解 17 / 官方动作画面

把动作定义交给系统

示例：Action type 配置



输入参数 → 业务规则 → 产生作用

读懂合同之后

还要知道当前允许做什么

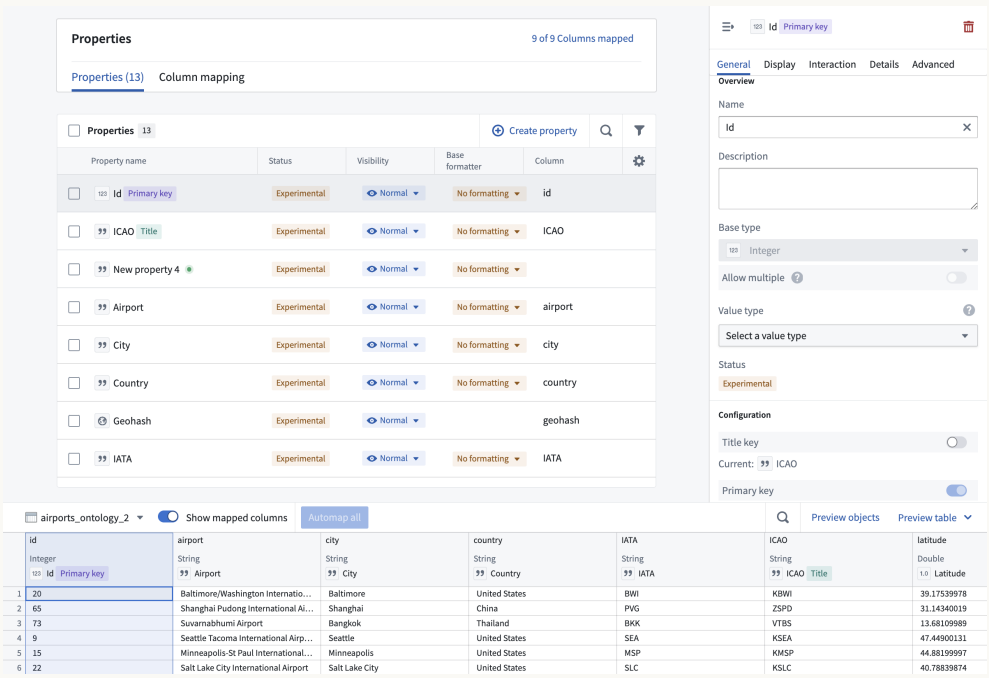
动作有输入和规则，不只是一句提示词。 官方示例展示机制，不是开票系统截图。

Palantir · Action type view · 官方示例动作

核心讲解 18 / 数据与共识

加一层模型
不会消除分歧

先确定：哪份记录可信



记录来源 · 对象映射 · 更新时间

权限与数据质量

仍需要系统和工程落实

保留记录来源与更新时间。 部门对可信记录没有共识，模型不能代替管理决定。

Palantir · Property editor · 官方产品截图

核心讲解 19 / 执行架构

Agent在上面
约束在执行中

原创逻辑架构 · 非厂商部署图

Agent

理解请求、补齐信息、提出处理计划

业务语义与动作定义

对象、关系、状态、证据与可用动作

工作流与受控接口

身份、权限、业务校验、审批、恢复与去重

CRM / 合同 / 项目 / ERP

保存业务记录，返回实际执行结果

原创逻辑架构：查询与动作边界均应执行授权。逻辑分层不等于物理部署边界。

底层是现有记录系统。 接口关联记录，语义层说明含义。 Agent提出计划，经过受控执行，再回写结果。

原创分析 · 教学示意

核心讲解 20 / 运行机制

任务要能
持续做下去

Workflow / Runtime

等待审批 → 收到结果后继续

持续保存任务状态

接口超时 → 先查是否已成功

恢复与去重，避免重复申请

合同变更 → 重新检查判断

不能继续沿用已失效的依据

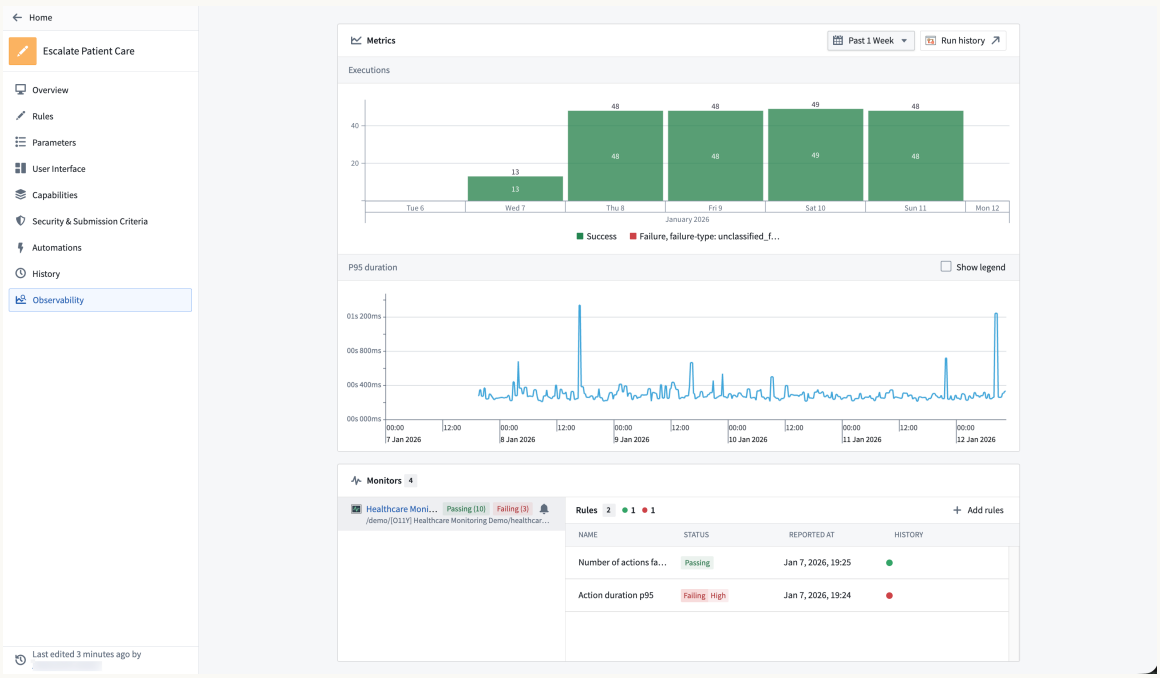
今天等待，明天继续。 接口超时不能盲目重复提交。 合同版本变化，原来的判断可能失效。

原创分析 · 教学示意

核心讲解 21 / 官方运行画面

执行过了
也要查得到

示例：Action observability



执行记录 · 延迟 · 运行状态

权限与状态限制

不能只写在提示词里

执行要留下记录，才能核验。 限制需要系统执行，不能只靠提示词。

Palantir · Action observability · 官方产品截图

核心讲解 22 / 实施路径

先做一段
能验收的业务

最小业务闭环 · 小范围不等于只做摘要

触发

发现付款节点可能满足条件

处理

核对证据，提交申请

结果

确认系统已接收，异常有人接手

从明确触发开始。 核对证据与提交申请。 验收时看得见业务推进了没有。

原创分析 · 教学示意

核心讲解 23 / 同步验证

边设计
边进入真实工作

FDE · 把业务要求变成能运行的实现

明确基线

负责人 / 时间 / 返工

业务设计

规则 / 数据 / 动作

工程验证

现场观察 / 异常测试

调整再验证

资料拿不到，就回查方案

同一个试点范围内

业务与工程反复对齐

记下原来的时间和返工。 FDE进入现场，看员工为什么修改和核对。 发现数据、规则问题，回来调整并再验证。

原创分析 · 教学示意

核心讲解 24 / 验收与演进

提效
要算完整的账

先辅助，再逐步开放经过验证的动作

任务做对了吗

对象、证据、授权、实际结果

整段工作变好了吗

等待、处理、核对与返工

经营有没有受益

观察回款变化，计入运行与维护成本

先检查任务做对了没有。 提交快但核对翻倍，要重新算账。 经营结果需观察与对照，改动后还要回归检查。

原创分析 · 教学示意

核心讲解 25 / 从管理到执行

一笔完整生意
一套可执行判断

下次评审 Agent 项目，先看这三件事

卡点的依据

项目究竟准备解决什么

工作的改变

谁负责，规则怎样落地

实际的结果

演示系统完成了什么

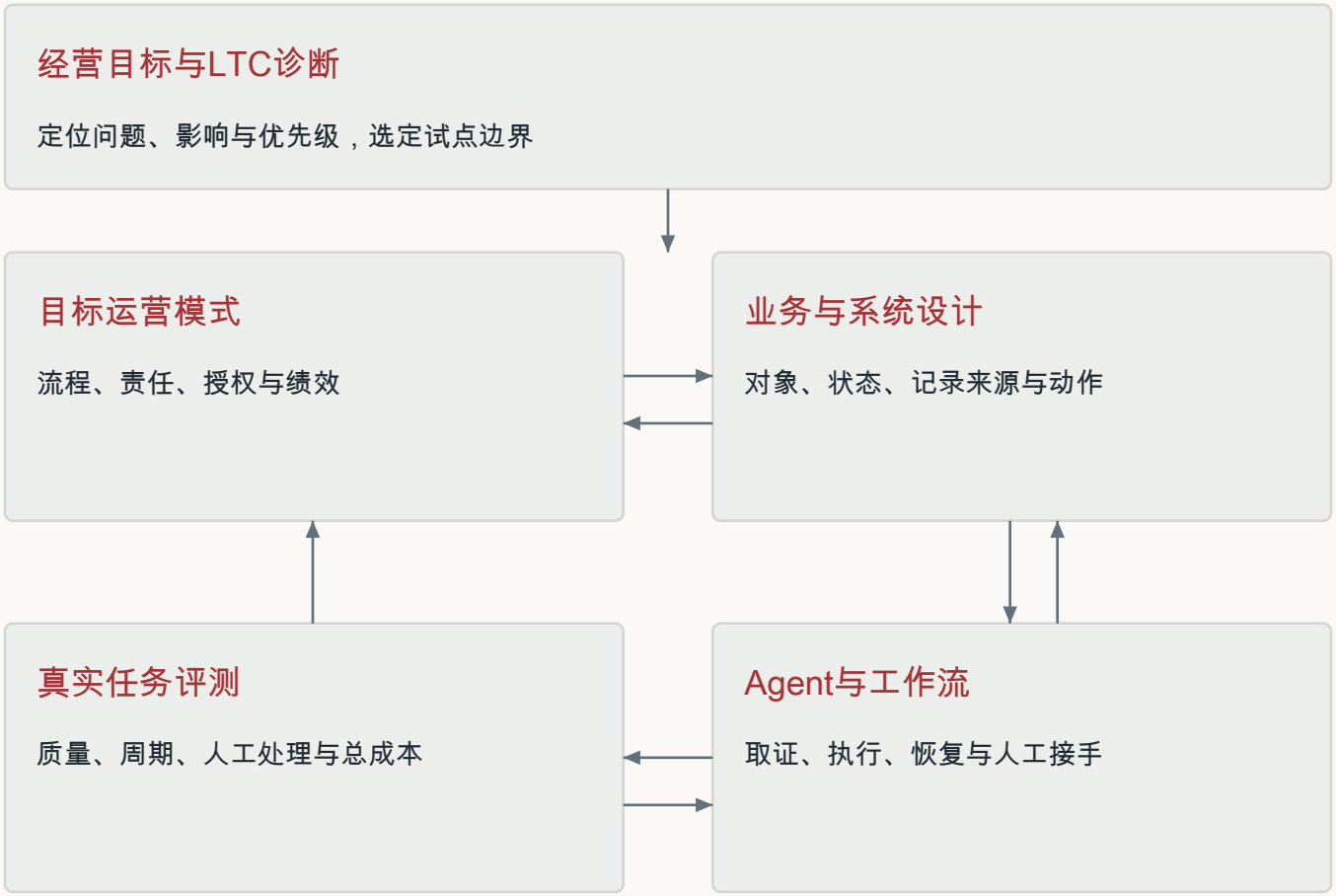
拿出业务卡点的依据。 说明改变哪些工作、谁负责。 演示实际结果，才有依据判断值不值得投。

原创分析 · 教学示意

原创架构图 / 01

在同一业务范围内共同验证

管理设计和工程验证来回推进，避免先规划完一切再试用。



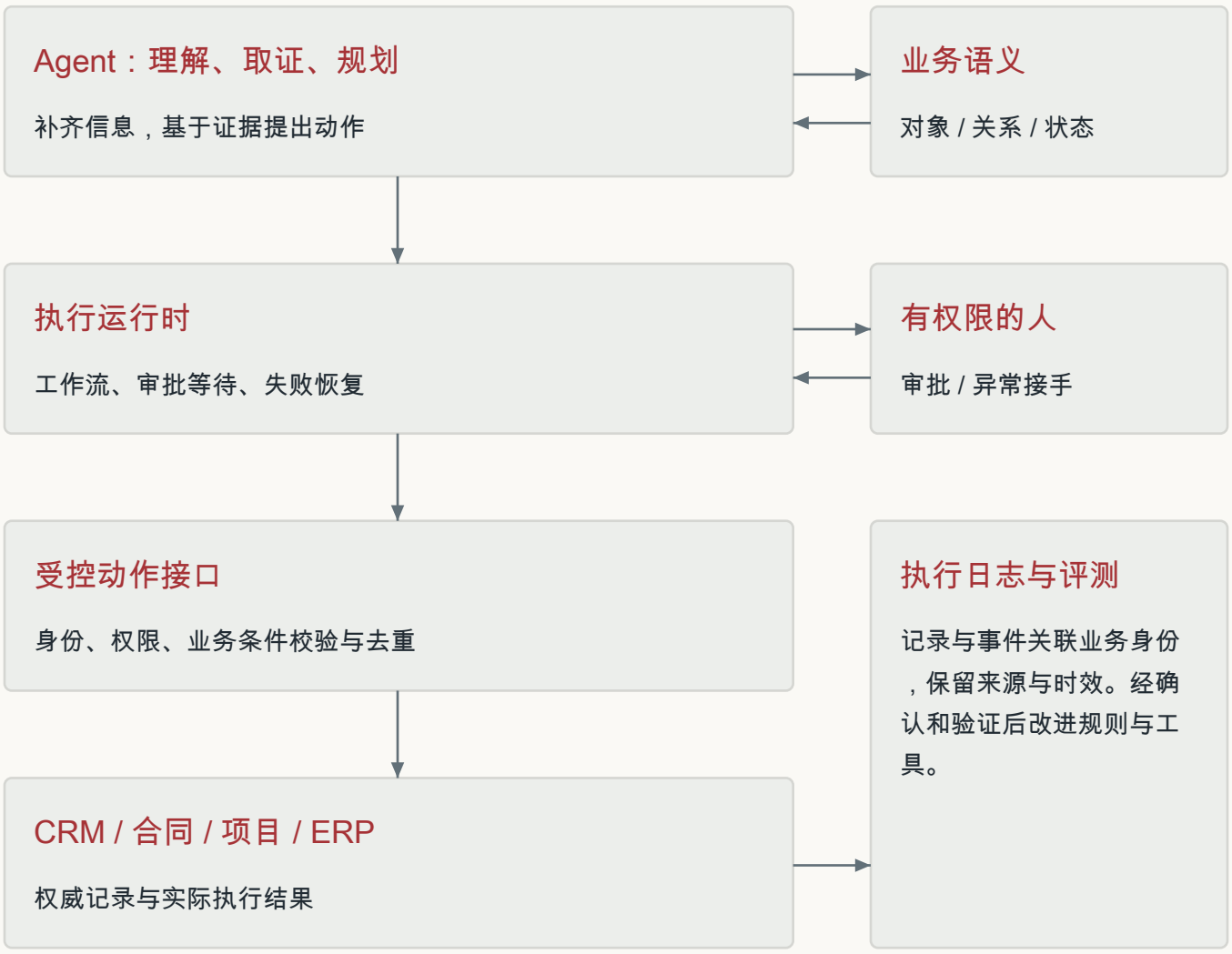
验证规则与职责可行后再扩大范围；新的异常继续回到诊断。

Ontology按跨系统语义复杂度与复用需要选择，并非每个试点的必选平台。企业架构贯穿业务与系统设计。此图为作者综合教学框架。

原创架构图 / 02

Agent参与LTC的逻辑执行架构

业务请求或事件触发；系统中的实际结果才是完成依据。



逻辑分层不等于部署边界。查询和动作均需授权；本体不自动承担审批、事务一致性或全部权限控制。跨系统失败需按业务可逆性设计恢复与补偿。

完整讲稿 / 01

Agent时代，企业为什么要重新学华为LTC？

企业做AI，为什么还需要懂管理？今天我们从管理咨询的视角，讲一套华为广泛应用的业务流程：LTC，从线索到回款。

它管的是一笔生意怎样从客户需求开始，经过销售、签约、交付，最后收到钱。

到了Agent时代，这套思路尤其值得重新理解。因为AI开始参与执行以后，公司需要讲清楚：依据什么判断，谁有权决定，怎样才算完成。

今天我把这件事从经营诊断讲到业务架构，再讲到Ontology和Agent怎样落地。让老板知道该投哪里，业务负责人知道该改什么，技术团队知道该怎么实现。

我想先讲一个容易被忽略的地方：企业提出要做什么项目，不一定已经找对了问题。

比如回款慢，第一反应可能是做一个催收助手。但如果客户一直没有验收，多发几次提醒，作用可能很有限。

我会先把时间拆开看。以先交付、验收，再开票付款的项目为例：从交付到验收，时间花在哪里？验收以后，开票有没有耽搁？合同约定什么时候付款，现在是否已经逾期？不同的付款安排，要按实际合同来分析。

这样一拆，解决方向就清楚了。卡在验收，要查履约和证据；卡在开票，要查资料和内部处理。到了逾期阶段，还得看有没有争议，客户是否存在支付问题。

其中，有些适合让Agent核对材料、准备申请；有些需要负责人协调，或者重新作商业判断。

咨询里讲价值驱动分析，就是要把经营目标拆到可以干预的地方。再比较影响范围、损失和改善成本，排出投入顺序。最容易自动化的工作，未必最值得先做。

LTC的用处就在这里：沿着整笔生意追原因。财务端暴露的问题，根源可能在前面的销售承诺或交付环节。

完整讲稿 / 02

从管理判断到系统执行

完整讲稿 · 续

从经营视角，我们可以把它看作一条价值流。价值流关心客户怎样得到价值，企业怎样实现相应的商业结果。

华为公开的LTC架构包括线索管理、机会点管理和合同履行管理。[1] 简单展开，就是判断需求是否值得跟进，形成方案和商业条件，再把合同约定的事情做完，完成相应收款。

合同评审、资源申请、开票申请，则是里面更具体的流程，规定由谁、在什么条件下完成哪些工作。

把这两个层次分开，就能理解：业务要从头看到尾，系统却可以分工。

CRM管机会，合同系统管条款和评审，项目系统管资源与交付，ERP承接相应的财务记录。具体怎么划，要看企业已有的系统和业务特点。

企业架构做的，就是把这种关系理清楚。业务准备怎么运行，数据怎么定义，应用各自负责什么，技术如何支撑，都要对得上。

但不需要每次都从零画一套宏大蓝图。既有系统已经有成熟功能，就先看能否利用；真正缺的部分，再决定怎么补。

接下来还有一个管理问题：这些工作，由谁来组织？

这涉及目标运营模式，Target Operating Model。你可以理解为，公司准备采用什么样的流程、分工和决策机制来经营，并用什么指标判断效果。

销售想尽快拿单，交付担心资源不够，财务关心新增成本。大家的考虑都有道理，公司需要有人能对这种取舍作决定。

所以，流程负责人不能只是一个负责催进度的人。他要推动跨部门问题的解决；超出权限的事情，也要有明确的裁决路径。

完整讲稿 / 03

从管理判断到系统执行

完整讲稿 · 续

考核同样要配合。如果销售只看签约金额，交付只看自己部门的成本，双方可能都完成了指标，项目却没做好。需要让相关负责人共同关注承诺是否兑现，同时保留各岗位能够控制的指标。

这些安排确定以后，才有依据设计每个阶段的推进条件、授权和例外处理。否则，把模糊的分工交给Agent，它还是得不停找人确认。

不过，管理规则讲清楚了，AI也未必能直接使用。它还需要理解系统里的业务。

一笔生意，在CRM里有机会编号，在合同系统里有合同号，交付时又有项目号。员工常常知道这些记录怎么对应，Agent却不能靠名称相似就认定它们是一回事。

这时就会涉及Ontology，也就是本体：用明确的业务对象和关系，组织大家对业务的共同理解。

知识库帮助找到材料，API提供查询和操作入口。本体可以进一步说明：这份合同对应哪些项目，某个付款节点需要哪些依据，“完成”究竟指交付结束还是客户已经验收。

这些能力可能集成在同一平台里，不一定要另建一套系统。简单场景，现有数据模型就可能够用；多个应用反复解释同一批对象，跨系统状态经常对不上，统一业务语义才更有价值。

还是拿开票来说，Agent要找到对应的合同和付款节点，检查约定条件，引用相关证据，再发起申请。有预付款的合同，就不能套用“先验收再开票”的规则。

所以，让AI读懂合同之后，还得让它知道，这份合同在当前状态下允许做什么。

Palantir的Ontology实现，把对象、关系和动作、安全控制等能力连接起来，可以作为参考。[2]但一张本体模型不会自动解决数据冲突，权限也需要具体系统来执行。

如果几个部门对哪份记录可信都没有共识，多加一层模型，并不会让分歧消失。

把这些内容放进架构图，Agent的位置就比较清楚了。

底层还是企业现有的CRM、合同、项目和财务系统，保存相应的业务记录。

完整讲稿 / 04

从管理判断到系统执行

完整讲稿 · 续

通过数据和事件接口，把记录关联起来，保留来源和更新时间。业务语义层负责说明它们的含义，以及可以调用哪些动作。

Agent读取这些信息，理解请求、补齐缺项、提出处理计划。执行时，要经过相应的权限和业务条件校验；需要审批的，就交给有权限的人。结果再写回负责这件事的系统。

这中间还有工作流和运行时，负责让任务持续进行。今天等审批，明天收到结果后能继续；接口超时，要先查提交是否成功，不能重试一次就多建一笔申请。

合同版本变了，原来的判断也可能失效，需要重新检查。

这些机制决定了Agent能不能用于日常业务。权限和状态限制，不能只靠一句提示词“请遵守规则”。

真正落地时，我会先选一个最小业务闭环。范围可以小，但要从触发一直走到可检查的结果。

例如，从发现某个付款节点可能满足条件，到核对证据、提交开票申请，并确认系统已经接收。这样，验收时看得见业务到底推进了没有。

先确定负责人，记下原来的处理时间和返工情况。然后围绕这一段，把规则、数据和动作接口一起做出来。不需要先完成全公司的架构设计或数据治理。

FDE在这里的价值，是进入实际工作现场，把业务要求变成能运行的实现。比如员工为什么总要改掉一条建议，为什么还要去另一套系统核对，要弄清原因，再调整规则或工具。

发现数据拿不到，就检查方案是否可行；发现同一个状态有不同解释，就请业务负责人确认。业务设计和工程验证，需要这样来回推进。

测试也要用真实任务。顺利的情况要看，资料缺失、版本变化、权限不足这些异常也要看。先让Agent辅助处理，再逐步开放经过验证的动作。出错以后，必须有人能接着处理。

最后验收，先检查任务本身有没有做对，再看整段工作是否改善。

完整讲稿 / 05

从管理判断到系统执行

完整讲稿 · 续

如果申请提交快了，但员工核对时间翻倍，就得重新算这笔账。等待、人工处理、返工，以及系统运行和维护成本，都要考虑。

至于回款有没有改善，要继续观察，并与相似业务或原来的基线比较。不能把局部省时直接算成经营收益。

后续修改规则、工具或模型，也要用历史任务检查，避免修好一个问题又引入另一个。企业经验经过这样的验证，才逐渐成为能复用的组织能力。

所以我建议做企业AI的人重新看LTC。它能帮助我们从一笔完整的生意出发，把管理上的判断，一直落实到系统怎么执行。

下次评审Agent项目，可以让团队拿出业务卡点的依据，说明准备改变哪些工作、谁来负责，再演示系统实际完成了什么。

如果这几件事说得清楚，老板就有依据判断值不值得投，业务和技术也更容易围绕同一个目标推进。

工作表 01

业务诊断卡

可直接用于团队讨论；以下空白用于填写自己的业务。

经营问题

用业务事实描述问题，避免直接把“做一个Agent”当成需求。

影响与基线

涉及多少任务？等待、人工处理、返工分别多久？依据来自哪里？

定位卡点

卡在销售承诺、合同、交付、验收、开票还是收款？按具体业务路径分析。

责任与优先级

谁能决定改法？改善价值、实施成本、持续维护成本是多少？

工作表 02

Agent动作定义卡

可直接用于团队讨论；以下空白用于填写自己的业务。

对象与依据

对象ID、关联关系、来源系统、更新时间、合同版本。

允许做什么

动作名称、输入参数、前置条件、查询与写入授权。

谁来决定

自动处理范围、审批人、超限升级路径、异常接手人。

怎样确认完成

目标系统返回值、业务状态、唯一请求标识、重试与去重、变更后的重新校验。

工作表 03

试点验收清单

可直接用于团队讨论；以下空白用于填写自己的业务。

业务与任务结果

是否实际完成目标动作？是否满足业务规则？有没有遗漏证据或越权？

异常覆盖

资料缺失、状态冲突、版本变化、权限不足、接口超时、重复提交。

完整成本

等待时间、人工核对、返工、模型与接口运行费用、维护投入。

扩大范围的依据

与原基线或相似业务比较；明确人工接手；规则、工具、模型变更后进行回归检查。

参考观点

从陈果的三篇文章继续往下想

以下为阅读归纳与本资料的延伸分析，不是作者原话。

LTC、数据与系统边界

先检查业务流程、记录来源和系统职责，再决定如何治理信息。本资料进一步关注：当Agent参与执行，业务对象、判断依据与可用动作怎样保持一致。

流程是组织的肌肉记忆

值得学习的是持续变革、知识沉淀和组织实践。本资料把它延伸为：从真实任务中发现异常，由负责人确认改动，再用历史任务验证效果。

4A的适用条件

架构工作应尊重企业实际与既有软件资产。本资料采用的判断是：成熟能力先复用，差异化部分再设计；不能把4A一概理解为仅适用于自研。

几个容易混淆的边界

数据治理可以包括标准、责任和源头控制，不只是清洗。流程可以包含人工任务，不必都能自动执行。本体可以支持业务语义，不能自动替代审批、权限和事务控制。

资料来源

官方文档与延伸阅读

来源用于学习与核验；原图版权归原作者或机构所有。

华为《网络安全透视》（2013），图2 / 印刷页13

https://www-file.huawei.com/-/media/corporate/pdf/cyber-security/hw-cyber-security-wp-2013-cn.pdf

Palantir：Ontology overview

https://www.palantir.com/docs/foundry/ontology/overview

Palantir：Ontology Manager overview（界面截图）

https://www.palantir.com/docs/foundry/ontology-manager/overview

IBM：What is data governance?

https://www.ibm.com/think/topics/data-governance

OMG：BPMN 2.0

https://www.omg.org/spec/BPMN/2.0/PDF

陈果：《LTC解决方案 | 数据、流程、企业架构和本体》

陈果：《学习华为转型经验 | 业务流程是组织的肌肉记忆》

陈果：《滥用华为4A架构，把很多中国企业的数字化带进沟里》

陈果文章依据用户提供正文做阅读归纳，未将其个人经历改写为本人的项目经验。官方图片来源记录于配套sources.json。