面向智能体决策的沙盘重构工程方案书

陈文轩，2026/04/week4 组会报告

一　引言

二　智能体总体架构设计综述（2026/04，week1–3 组会文件整理）

课题组设计的智能体系统采用多 Agent 协同架构，核心包含五大智能体角色，通过决策后的数据流通，形成完整的决策闭环。系统架构如图1所示，主要智能体及其职能定义如下：

1. **商分 Agent**：负责解析商业环境，计算并输出当前年度的**商分系数（商业环境权重）**。其输入包括比赛规则、市场详单以及竞争对手信息，是整个决策链路的起点。
2. **决策 Agent**：接收商分 Agent 输出的环境权重，结合知识库（RAG）检索优秀历史决策案例，生成若干备选经营方案，并负责在后续阶段进行策略微调。
3. **PSS Agent**（Product-System-Strategy Agent）：面向企业内部的经营单元进行建模与计算。PSS 是系统的核心决策单元，按“产品—市场—生产线”三维识别，并可按市场、产品、生产线及其组合进行聚合，负责计算各单元的价值密度（VPD）、运营成本（OE）以及内部耦合系数。
4. **EPSS Agent**（Enterprise-level PSS Agent）：在企业全局层面聚合各 PSS 单元的计算结果，计算企业级的综合耦合指标与方案打分，并调用现金流工具验证方案的可行性。
5. **选单 Agent**：在方案确定后，负责调用广告与选单工具，执行具体的市场投放与订单选择操作。

此外，系统还配备了一系列外部工具集，包括排产工具、详单工具、现金流工具、报表工具、竞单分析工具等，为各 Agent 提供数据中转与计算转化能力。

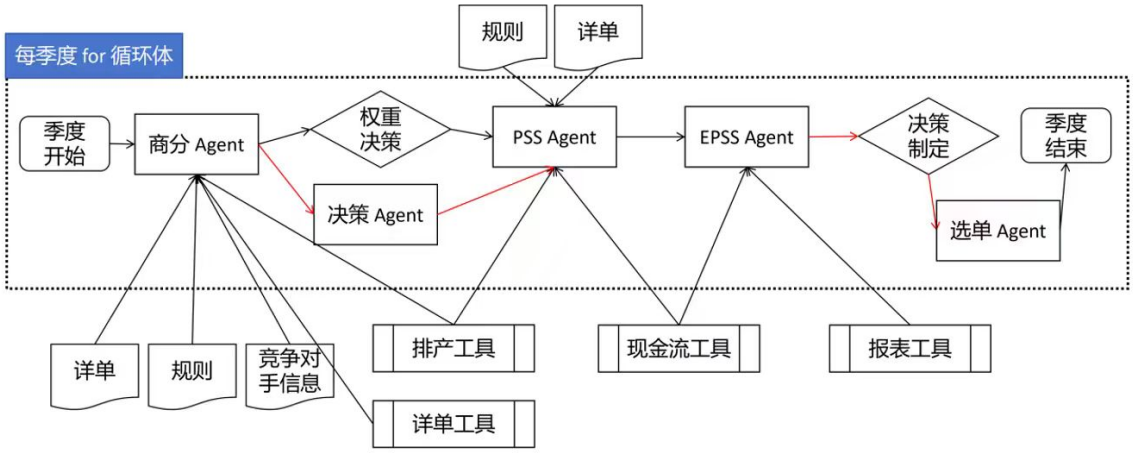

图 1: 多智能体协同决策系统总架构图

2.1 商分 Agent 设计综述与个人改进建议

在多 Agent 协作框架中，商分 Agent 承担着整个决策链路的“感知—认知”职能。其根本任务并非直接排产或生成采购计划，而是在年度初抢单前，将历史详单、静态规则以及年初巡盘数据转化为本年度的市场权重、产品权重、产线权重以及产品—市场、产品—产线组合优先级，为**后续决策 Agent 与选单 Agent 提供量化的环境参数**。

课题组为商分 Agent 设计了“四层递进”计算架构，即**先验分析层、巡盘修正层、权重融合层与反馈更新层，形成“先验—修正—输出—再学习”的完整闭环**。

2.1.1 先验分析层：基于历史规律与规则的初始权重生成

先验分析层**仅依赖历史详单与静态规则**，回答“在不考虑本年度竞争态势的前提下，哪些市场、产品与产线组合先天更值得投入”，在此分析层，不考虑。该层构建了四类先验评分模型：

（1）产品先验分

产品先验分**综合需求强度、增长趋势、价格强度与账期压力四项子指标：**

$$S_p^{\text{prior}}(y) = a_1 \cdot D_p(y) + a_2 \cdot G_p(y) + a_3 \cdot P_p(y) - a_4 \cdot A_p(y) \tag{1}$$

其中，$D_p(y)$ 为需求强度（由历史详单组均需求量归一化得到），$G_p(y)$ 为增长趋势（相邻年度需求增长率），$P_p(y)$ 为价格强度（历史平均单价），$A_p(y)$ 为账期压力（加权平均账期）。初始参数取 $a_1 = 0.4, a_2 = 0.2, a_3 = 0.3, a_4 = 0.1$。

（2）市场先验分

市场先验分衡量各细分市场的投入价值：

$$S_m^{\text{prior}}(y) = b_1 \cdot D_m(y) + b_2 \cdot \text{ROI}_m + b_3 \cdot \text{Fit}_m(y) - b_4 \cdot \text{EntryCost}_m \tag{2}$$

其中，ROI_m 为**市场投入收益（市场分值与开发成本之比）**，$\text{Fit}_m(y)$ 为**市场适配度（ISO 资格与历史表现的综合）**，EntryCost_m 为**进入成本惩罚**。初始参数取 $b_1 = 0.35, b_2 = 0.25, b_3 = 0.25, b_4 = 0.15$。

（3）产品—市场先验分

产品—市场先验分是商分 Agent 输出的最关键先验量，直接指导后续 Agent “某产品应投放到哪一市场”：

$$S_{pm}^{\text{prior}}(y) = c_1 D_{pm}(y) + c_2 P_{pm}(y) + c_3 \text{Fit}_{pm}(y) - c_4 A_{pm}(y) - c_5 \text{EntryCost}_m \tag{3}$$

各子项均从详单表动态计算：$D_{pm}(y)$ 为该产品在该市场的历史成交量，$P_{pm}(y)$ 为历史均价，$\text{Fit}_{pm}(y)$ 综合 ISO 匹配度与历史成交表现，$A_{pm}(y)$ 为加权平均账期。初始参数取 $c_1 = 0.35, c_2 = 0.25, c_3 = 0.15, c_4 = 0.10, c_5 = 0.15$。

（4）产品—产线先验分

产品—产线先验分用于评估 “**某产品由哪类生产线承接更优**”：

$$S_{pl}^{\text{prior}}(y) = d_1 \text{Margin}_{pl}(y) + d_2 \text{SpeedFit}_{pl}(y) + d_3 \text{Flex}_l - d_4 \text{InvestBurden}_l - d_5 \text{ConvCost}_l \tag{4}$$

其中，$\text{Margin}_{pl}(y)$ 为粗利润空间（均价减直接成本），$\text{SpeedFit}_{pl}(y)$ 为交付紧迫度与线型速度的匹配度，Flex_l 为柔性加分，InvestBurden_l 与 ConvCost_l 分别为投资负担与转产负担。初始参数取 $d_1 = 0.30, d_2 = 0.25, d_3 = 0.15, d_4 = 0.15, d_5 = 0.15$。

上述四类先验分经 Softmax 归一化后，得到产品、市场、产线的初始权重向量 w_p^{prior}、w_m^{prior} 与 w_l^{prior}。

2.1.2 巡盘修正层：基于竞争对手布局的动态修正

巡盘修正层的核心思想是：**不在先验分之外重建一套全新评分体系，而是利用年初可获取的竞争对手巡盘数据，对先验权重进行定向修正**。该层回答的问题是—— “在看见对手真实的生产线布局与广告投放后，原有基于历史规律的判断应如何调整”。修正机制按数据来源分为两类：ProductLine 修正（修正产品权重与产品—产线优先级）和 AD 修正（修正市场权重与产品—市场优先级）。两类修正的对比如表1所示。

表 1: 巡盘修正层两类修正机制对比

对比维度	ProductLine 修正	AD 修正
修正对象	产品权重、产品—产线优先级	市场权重、产品—市场优先级
数据来源	ProductLine.xls（生产线信息）	AD.xls（广告投放矩阵）
核心指标	供给压力 SupplyPress、单队竞争强度 PerCompCap、现实线型偏好 LinePref、自身匹配度 SelfFit、转产风险 ConvRisk	广告压强 ADPress、单队平均广告 AvgAdSpend、最大对手广告 ADMax、广告集中度 ADConc、预计拿单概率 OrderProb
输出修正量	ΔS_p^{PL}、$\Delta S_{pl}^{\mathrm{PL}}$	$\Delta S_{pm}^{\mathrm{AD}}$
参数建议	$e_1 = 1.0$; $f_1 = 0.45, f_2 = 0.35, f_3 = 0.20$	$g_1 = 0.40, g_2 = 0.35, g_3 = 0.25$

（1）ProductLine 修正——从竞争供给到产线选择

ProductLine 修正以竞争对手的生产线信息表为输入，其修正逻辑遵循“识别竞争烈度 → 评估现实偏好 → 叠加自身匹配”三步映射，具体流程如图2所示。

1. **竞争烈度识别**：首先统计实际持有某产品产线的对手队伍数 $\mathrm{ActiveTeams}_p(y)$，进而计算两项互补指标——供给压力 $\mathrm{SupplyPress}_p(y)$（所有对手该产品产线能力之和的归一化值）与单队竞争强度 $\mathrm{PerCompCap}_p(y)$（人均产线能力）。两者配合使用可区分两种典型竞争格局：SupplyPress 高且 PerCompCap 低，对应“蜂群竞争”（人多但人均不强）；SupplyPress 低且 PerCompCap 高，对应“精英竞争”（人少但人均极强）。后者对己方的威胁往往更大，是修正中需要重点降权的情形。
2. **现实偏好评估**：通过统计对手使用各线型生产某产品的次数 $\mathrm{UseCount}_{pl}(y)$，计算产品—产线现实偏好 $\mathrm{LinePref}_{pl}(y)$，反映“市场上大多数人认为某产品更适合哪类产线”的群体智慧。
3. **自身匹配叠加**：引入自身匹配度 $\mathrm{SelfFit}_{pl}(y)$（综合己方已有线型规模、可快速投入线型数量及闲置风险）与转产风险 $\mathrm{ConvRisk}_l(y)$（转产费用与周期的归一化值），在群体偏好基础上叠加企业自身的资源禀赋约束。

上述三步映射最终收敛为两项修正量：

$$\Delta S_p^{\mathrm{PL}}(y) = -e_1 \cdot \mathrm{SupplyPress}_p(y) \tag{5}$$

$$\Delta S_{pl}^{\mathrm{PL}}(y) = f_1 \cdot \mathrm{LinePref}_{pl}(y) + f_2 \cdot \mathrm{SelfFit}_{pl}(y) - f_3 \cdot \mathrm{ConvRisk}_l(y) \tag{6}$$

其中，产品修正量 ΔS_p^{PL} 为负向惩罚——竞争越激烈，产品权重越应下调；产品—产线修正量 $\Delta S_{pl}^{\mathrm{PL}}$ 则为正向引导——现实偏好高、自身匹配好且转产风险低的组合应获得加权。

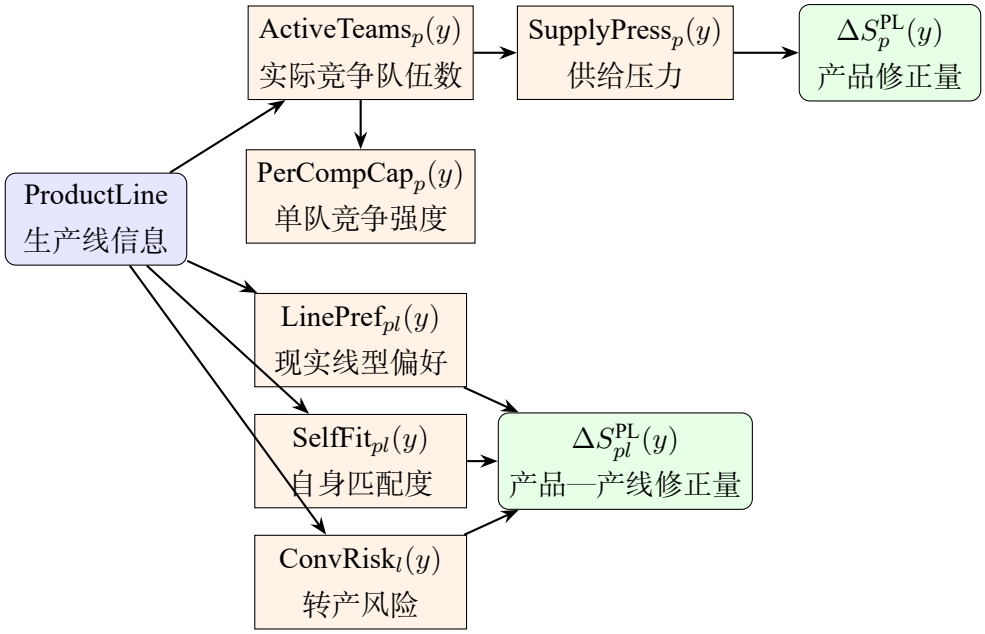

图 2: ProductLine 修正映射流程

（2）产品修正量到先验分四维权重的映射机制

由式 (5) 得到的产品修正量 $\Delta S_p^{\mathrm{PL}}(y)$ 并非仅作为一个总分惩罚项简单叠加，而是被进一步**映射**到产品先验分的四个子项权重 $\{a_1, a_2, a_3, a_4\}$ 上，实现“竞争压力 → 策略侧重”的精细调节。课题组设计的映射逻辑如表2所示，其核心假设是：竞争越激烈，企业越应从“追量”转向“追质”，即从单纯依赖需求规模转向更加重视增长潜力、价格利润与资金回笼速度。

表 2: 竞争压力到产品先验分四维权重的映射示例

先验子项	原权重	映射系数	调节方向	经济学含义
需求强度 $D_p(y)$	$a_1 = 0.40$	$\lambda_1 = -0.15$	↓下调	竞争者众，僧多粥少，单纯需求量大不足以支撑高权重
增长趋势 $G_p(y)$	$a_2 = 0.20$	$\lambda_2 = +0.10$	↑上调	拥挤市场中，仅增长趋势好的产品值得逆势投入
价格强度 $P_p(y)$	$a_3 = 0.30$	$\lambda_3 = +0.10$	↑上调	竞争加剧压缩利润，必须选择单价高、利润空间大的产品
账期压力 $A_p(y)$	$a_4 = 0.10$	$\lambda_4 = +0.05$	↑上调	对手可能放宽账期抢单，必须更重视资金回笼速度

映射函数采用线性调节形式：

$$a_i^{\text{修正}}(y) = a_i + \lambda_i \cdot \left|\Delta S_p^{\mathrm{PL}}(y)\right|, \quad i \in \{1, 2, 3, 4\} \tag{7}$$

其中 λ_i 为各维度的映射系数（见表2），$\left|\Delta S_p^{\mathrm{PL}}(y)\right|$ 为竞争压力的绝对强度。该映射

保证：当市场竞争趋于白热化时，产品先验分的内部结构发生“从量到质”的权重迁移——a_1（需求强度）被压缩，而 a_2, a_3, a_4（增长、价格、账期）被放大，从而使先验评分体系更贴合高竞争环境下的生存逻辑。上述映射流程如图3所示。

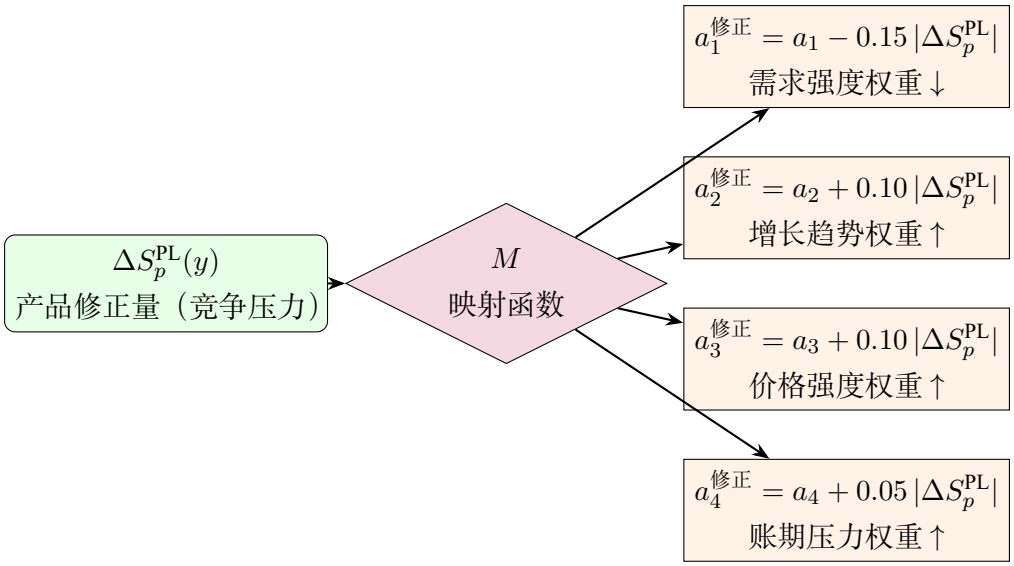

图 3: 产品修正量到先验分四维权重的映射流程

（3）AD 修正——从广告博弈到拿单概率

AD 修正以竞争对手的广告投放矩阵为输入，其修正逻辑遵循“**识别广告战烈度 → 判断竞争格局 → 估算拿单概率**”三步映射，具体流程如图4所示。

1. **广告战烈度识别**：首先统计在产品—市场上实际投放广告的对手队伍数 $\mathrm{ActiveTeams}_{pm}(y)$，进而计算广告压强 $\mathrm{ADPress}_{pm}(y)$（所有对手广告额之和的归一化值）与单队平均广告投入 $\mathrm{AvgAdSpend}_{pm}(y)$。与 ProductLine 修正类似，两者配合可区分“分散式竞争”（人多但每人投得少）与“精准打击”（人少但每人投得重），后者需要针对性超越单一最强对手，策略截然不同。
2. **竞争格局判断**：计算最大对手广告额 $\mathrm{ADMax}_{pm}(y)$ 及广告集中度 $\mathrm{ADConc}_{pm}(y) = \mathrm{ADMax}/\mathrm{ADPress}$。$\mathrm{ADConc}_{pm}$ 作为比值指标，不受队伍总数影响，可准确识别“一家独大”（集中度高）与“充分分散”（集中度低）两种格局。
3. **拿单概率估算**：综合历史中标率 $\mathrm{HistWin}_{pm}$、硬性门槛通过度 $\mathrm{GatePass}_{pm}$（市场资格、ISO 等）与广告差距 AdGap_{pm}，估算预计拿单概率 $\mathrm{OrderProb}_{pm}(y)$，作为正向激励项纳入修正。

上述三步映射最终**收敛为产品—市场修正量**：

$$\Delta S_{pm}^{\mathrm{AD}}(y) = g_1 \cdot \mathrm{OrderProb}_{pm}(y) - g_2 \cdot \mathrm{ADPress}_{pm}(y) - g_3 \cdot \mathrm{ADConc}_{pm}(y) \tag{8}$$

该式的经济学含义十分清晰：$\mathrm{OrderProb}_{pm}$ 越高，说明拿单胜算越大，应正向加权；$\mathrm{ADPress}_{pm}$ 与 ADConc_{pm} 越高，说明广告战越激烈或头部垄断越强，应反向惩罚。三项

的权重配比 $g_1 : g_2 : g_3 = 0.40 : 0.35 : 0.25$，体现了“机会优先、风险次之、格局再次”的稳健修正策略。

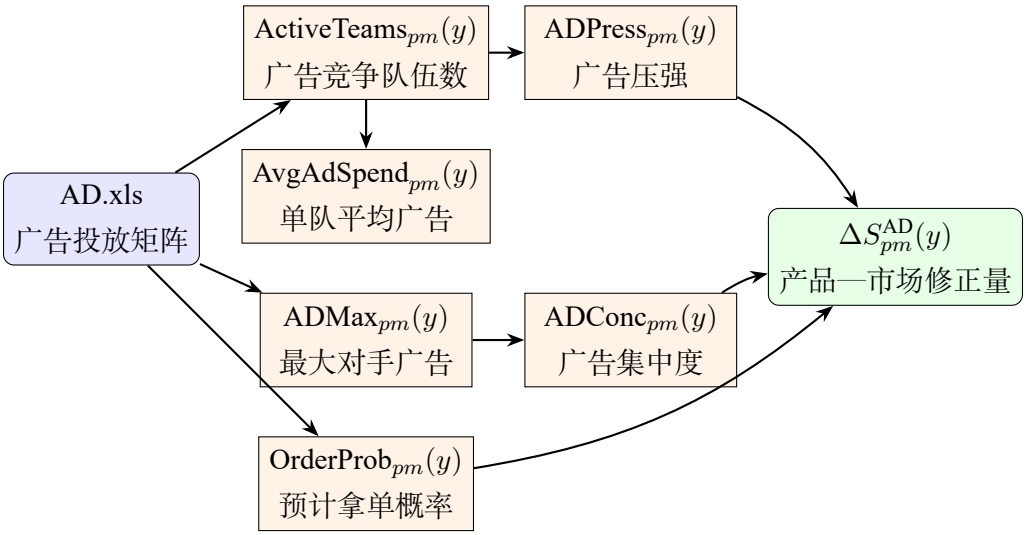

图 4: AD 修正映射流程

2.1.3 权重融合层：多源评分到最终权重的聚合

权重融合层的任务是将先验分析层与巡盘修正层产生的多源评分聚合为一套统一的、可执行的权重体系。具体而言，该层需要融合三**类独立权重**（产品权重、市场权重、产线权重）和**两类组合优先级**（产品—市场优先级、产品—产线优先级）。

（1）融合对象与融合公式

权重融合涉及以下五个核心对象的最终评分计算：

1. **产品最终分**：**先验产品分与 ProductLine 产品修正量的直接叠加**（修正量本身为负向惩罚，故用加法）：

$$S_p^{\text{final}}(y) = S_p^{\text{prior}}(y) + \Delta S_p^{\text{PL}}(y) \tag{9}$$

2. **市场最终分**：先验市场分与 AD 市场修正量的叠加：

$$S_m^{\text{final}}(y) = S_m^{\text{prior}}(y) + \Delta S_m^{\text{AD}}(y) \tag{10}$$

其中 $\Delta S_m^{\text{AD}}(y)$ 由该产品在各市场上的 AD 修正量加权平均得到。

3. **产品—市场最终分**：先验组合分与 AD 组合修正量的加权融合，融合系数为 α：

$$S_{pm}^{\text{final}}(y) = \alpha \cdot S_{pm}^{\text{prior}}(y) + (1-\alpha) \cdot \Delta S_{pm}^{\text{AD}}(y) \tag{11}$$

4. **产品—产线最终分**：先验组合分与 ProductLine 组合修正量的加权融合，融合系数为 β：

$$S_{pl}^{\text{final}}(y) = \beta \cdot S_{pl}^{\text{prior}}(y) + (1-\beta) \cdot \Delta S_{pl}^{\text{PL}}(y) \tag{12}$$

建议取 $\alpha = 0.65, \beta = 0.60$，其经济学含义是：产品—市场的投放决策对实时竞争态势（广告博弈）更敏感，故先验占比略低（65%）；而产品—产线的产能配置对历史规律和自身禀赋的依赖更强，故先验占比更高（60%）。

（2）Softmax 归一化与最终输出

融合后的评分需经 **Softmax 归一化**，使各类权重在内部求和为 1，保证可比性与可执行性：

$$w_p^{\text{final}}(y) = \frac{\exp(S_p^{\text{final}}(y))}{\sum_{p'} \exp(S_{p'}^{\text{final}}(y))}, \quad w_m^{\text{final}}(y) = \frac{\exp(S_m^{\text{final}}(y))}{\sum_{m'} \exp(S_{m'}^{\text{final}}(y))}, \quad w_l^{\text{final}}(y) = \frac{\sum_p \exp(S_{pl}^{\text{final}}(y))}{\sum_l \sum_p \exp(S_{pl'}^{\text{final}}(y))} \tag{13}$$

权重融合层的最终输出为以下**五类参数**，直接供后续抢单与竞单工具调用：

1. 产品排序向量：$\{w_p^{\text{final}}(y)\}_{p=1}^{P}$
2. 市场排序向量：$\{w_m^{\text{final}}(y)\}_{m=1}^{M}$
3. 产线排序向量：$\{w_l^{\text{final}}(y)\}_{l=1}^{L}$
4. 产品—市场优先级矩阵：$\{S_{pm}^{\text{final}}(y)\}_{p,m}$
5. 产品—产线优先级矩阵：$\{S_{pl}^{\text{final}}(y)\}_{p,l}$

2.1.4 反馈更新层：从实际绩效到滚动学习

反馈更新层是商分 Agent 实现“自我进化”的关键闭环，只有实现反馈才能实现对于模型各个权重的训练与优化。该层在**年末执行完毕后**启动，将本年度实际经营结果转化为对下一年度权重的修正信号，形成“预测—执行—评估—再学习”的滚动迭代。

（1）反馈数据来源

反馈信号来源于三**类实际绩效数据**，这个是商分 Agent 系统，对于沙盘决策的实际结果反馈，是 Agent 智能化程度的体现：

1. **抢单与订单结果**：各产品—市场的实际中标率 $\text{WinRate}_{pm}(y)$、实际利润率 $\text{ProfitRate}_{pm}(y)$、广告浪费率 $\text{AdWaste}_m(y)$（广告投入/实际取得订单量）。
2. **PSS 健康度报告**：各 PSS 单元的价值密度 VPD、运营成本 OE、平均健康度 $\text{Health}_p(y)$（可用平均 VPD/OE 代表）。
3. **企业级耦合评价**：本年度耦合协调度 $H(y)$、资产增速 U_1、效率得分 U_2、公平得分 U_3，用于判断整体决策质量的协调水平。

（2）反馈运算机制

反馈运算**分为三条独立更新链路**，分别对应市场权重、产品权重与产线权重：

$$\hat{w}_m(y+1) = w_m^{\text{final}}(y) + h_1 \cdot \text{WinRate}_m(y) + h_2 \cdot \text{ProfitRate}_m(y) - h_3 \cdot \text{AdWaste}_m(y) \quad (14)$$

$$\hat{w}_p(y+1) = w_p^{\text{final}}(y) + i_1 \cdot \text{Profit}_p(y) + i_2 \cdot \text{Health}_p(y) - i_3 \cdot \text{Loss}_p(y) \quad (15)$$

$$\hat{w}_l(y+1) = w_l^{\text{final}}(y) + j_1 \cdot \text{Util}_l(y) + j_2 \cdot \text{Delivery}_l(y) + j_3 \cdot \text{Health}_l(y) - j_4 \cdot \text{ConvCost}_l(y) \quad (16)$$

其中, $\text{Loss}_p(y)$ 为库存损耗、转产损耗、违约损失等综合惩罚; $\text{Util}_l(y)$ 为线型利用率; $\text{Delivery}_l(y)$ 为交付率。建议参数分别为 $h_1 = 0.35, h_2 = 0.40, h_3 = 0.25$; $i_1 = 0.40, i_2 = 0.35, i_3 = 0.25$; $j_1 = 0.30, j_2 = 0.25, j_3 = 0.25, j_4 = 0.20$。

（3）平滑更新与归一化

为避免权重大起大落，采用指数平滑机制:

$$w(y+1) = (1-\mu) \cdot w(y) + \mu \cdot \hat{w}(y+1) \quad (17)$$

其中 μ 为学习率。平滑后的权重再经归一化:

$$w_i(y+1) = \frac{w_i(y+1)}{\sum_i w_i(y+1)} \quad (18)$$

（4）反馈闭环与参数更新

归一化后的权重 $w(y+1)$ 被写入**年度权重更新表**，作为下一年度先验分析层的**初始权重种子**。具体而言:

1. 市场权重 $w_m(y+1)$ 注入下一年市场先验分的基准偏移;
2. 产品权重 $w_p(y+1)$ 影响下一年产品先验分的初始分布;
3. 产线权重 $w_l(y+1)$ 作为产品—产线先验分的偏好先验。

由此，反馈更新层与先验分析层形成跨年度闭环，商分 Agent 在每一轮比赛中持续积累环境认知，实现策略的自适应演化。

2.1.5 商分 Agent 总架构：四层闭环总览

商分 Agent 的整体架构可概括为“四层递进、两路输入、五类输出、年度闭环”。图5以 Tikz 流程图形式展示了各层之间的输入、运算与输出关系。

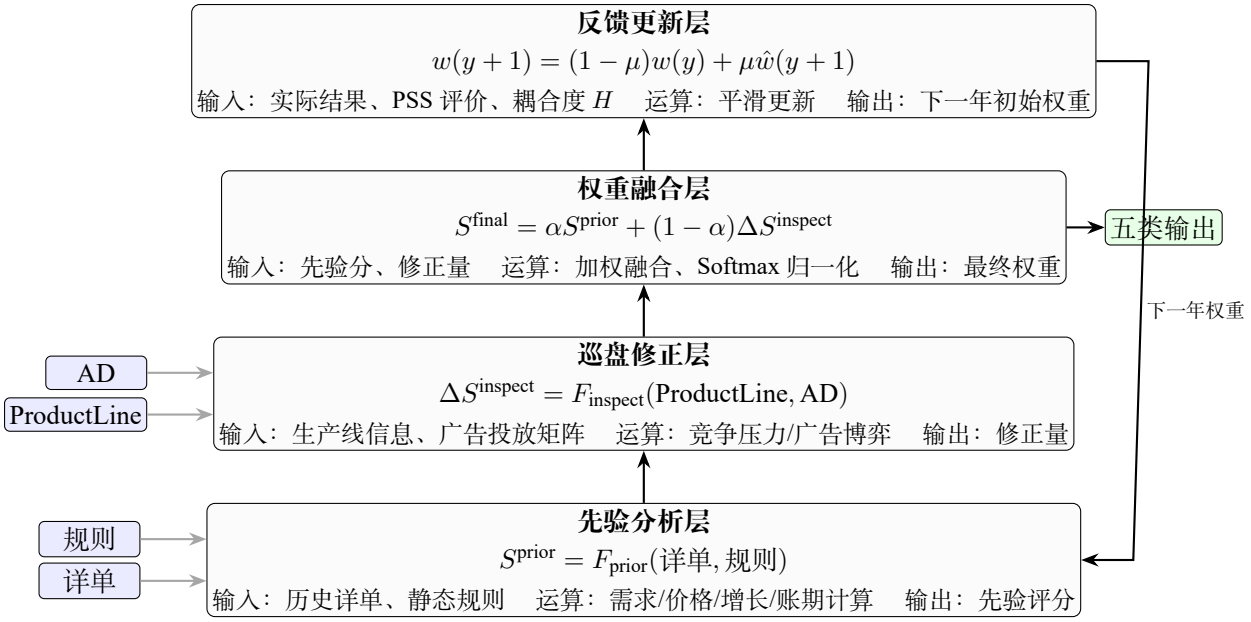

图 5: 商分 Agent 四层闭环总架构

2.1.6 商分 Agent 封装说明

对于调用商分 Agent 的其他开发者而言，可以内部四层运算细节没有那么详细的理解，只需遵循 ERPAI 协议（详见第3.4.1节）所定义的输入输出接口规范即可，这样的工程设计也是利用最终企业产品服务系统智能体的闭环落地，能够快速查找问题，并分析解决。

课题组在 4 月的《商分智能体完整方案》中已勾勒出先验—修正—融合—反馈的四层运算框架，并在 ReportPro 前文将其固化为可执行的公式体系。然而，随着工程化推进，以下三类结构性矛盾逐渐暴露，若不加以正视，将**直接影响商分 Agent 能否从” 纸面算法” 转化为” 可自适应演化的智能体”**。

2.1.7 学习率与收敛效率的结构性缺陷

当前方案在反馈更新层采用固定学习率 $\mu = 0.20$，即每年度保留 80% 的旧权重、吸收 20% 的新观测信号。该设定在理论上可避免权重大起大落，但与商业沙盘比赛的实际约束存在深层不匹配。

第一，年度轮次极度有限。标准沙盘比赛仅 5 ∼ 6 个年度，意味着商分 Agent 至多经历 4 ∼ 5 次权重更新。在 $\mu = 0.20$ 下，初始先验权重对最终分布的衰减系数为 $(1-\mu)^{n-1}$；即使经过 5 年迭代，第 1 年的初始偏见仍残留约 41% 的影响力。若初始专家参数 $a_1 = 0.40, b_1 = 0.35$ 等与实际环境严重错位，Agent 在整个比赛周期内都没有足够的机会将其” 洗出”。

第二，反馈信号本身的信噪比不确定。商分 Agent 的反馈输入（中标率、利润率、VPD/OE、耦合协调度 H）并非纯粹的” 权重质量” 度量，而是下游决策 Agent、选单 Agent、EPSS-Agent 共同作用后的**混合结果**。20% 的高学习率意味着商分 Agent 将大量

下游决策噪声当作自身权重误差来吸收，极易引发过拟合——即对某一年度的特殊竞争格局过度反应，反而破坏跨年度的一般性。

第三，固定学习率的改进方向——Warm-up 衰减策略。针对上述问题，笔者建议将固定学习率改进为 **Warm-up 衰减控制函数** $\mu(y)$：

$$\mu(y) = \begin{cases} \mu_0 \cdot \dfrac{y}{y_{\text{warm}}}, & y \le y_{\text{warm}} \\ \mu_0 \cdot \dfrac{1}{1 + k(y - y_{\text{warm}})}, & y > y_{\text{warm}} \end{cases} \tag{19}$$

建议取 $\mu_0 = 0.15$，$y_{\text{warm}} = 2$。前两年为 Warm-up 期，学习率从 0.075 线性增长至 0.15，避免初期因专家参数偏差过大而剧烈震荡；第三年起进入衰减期，逐步降低更新步长，防止末期过拟合。在工程实现中预留 μ_0、k、y_{warm} 三个超参数接口，供后续根据实际数据调优。

第四，更深层的局限——结构参数仍未开放。然而，即便引入 Warm-up 策略，学习率调节的仍是" 何时更新" 与" 更新多快"，却未回答" 更新什么"。当前反馈更新层仅调整最终权重向量 $w(y+1)$，而先验分析层中 a_i, b_i, c_i, d_i 等**结构参数**始终固定。若结构参数本身存在系统性偏差（例如专家给定的 0.40/0.20/0.30/0.10 产品权重配比并不符合当前比赛规则），仅对学习率做精细调节，无异于在错误的方向上优化步长。Warm-up 策略能够缓解收敛过程的震荡，却无法修正先验公式本身的结构性偏误，这是当前设计中最隐蔽却也最关键的缺陷。

2.1.8 自训练闭环的未完成性：验证迟缓与环境耦合

商分 Agent 最致命的架构缺陷在于：**它在单年度内无法完成自身的训练闭环，权重的合理性验证必须等待全系统执行完毕后才能进行，且未与具体执行环境解耦**。

（1）验证迟缓——年度级的反馈延迟。根据方案设计，反馈更新层" 在年末执行完毕后启动"，输入的三类信号——抢单结果、PSS 健康度报告、企业级耦合评价——全部来自下游 Agent 的年度终局产出。这意味着：

1. 年初输出的 $w^{\text{final}}(y)$ 在当年是**冻结的**，商分 Agent 在年度执行过程中无任何机制感知该权重是否有效；
2. 若权重输出存在方向性错误（例如因对手策略突变导致某产品权重被严重高估），该错误将无阻碍地传导至决策 Agent、选单 Agent，并在全年经营中不断放大；
3. 即使年末发现了错误，修正也只能作用于 $y+1$ 年，而沙盘比赛通常在第 $5 \sim 6$ 年即告结束，修正窗口极为狭窄。

这种" 年度级慢闭环" 与强化学习中 Episode-level 的更新节奏类似，但沙盘的特殊性在于：每个 Episode（年度）的代价极高——一次错误权重可能直接导致全年现金流断裂或市场份额丧失，且**没有重试机制**。

（2）未与环境解耦——商分 Agent 缺乏独立的输出质量评价指标。审视方案中四个

总式：

$$W^{\text{prior}}(y) = F_{\text{prior}}(\text{详单}, \text{规则}) \tag{20}$$

$$\Delta W^{\text{inspect}}(y) = F_{\text{inspect}}(\text{ProductLine}, \text{AD}) \tag{21}$$

$$W^{\text{final}}(y) = F_{\text{merge}}(W^{\text{prior}}, \Delta W^{\text{inspect}}) \tag{22}$$

$$W^{(y+1)} = F_{\text{update}}(W^{\text{final}}(y), \text{实际结果}, \text{PSS 评价}) \tag{23}$$

$W^{\text{final}}(y)$ 的输出质量究竟如何衡量？当前方案给出的答案是：**必须等实际结果出来才知道**。商分 Agent 没有一个内嵌的仿真器或评价函数来独立评估" 给定一组权重，预期经营绩效如何"。它把自身输出的验证完全外包给了后续 Agent 的物理执行过程——这正是" 未与环境解耦" 的核心含义。

一个更理想的架构应当是：商分 Agent 内部维护一个**环境模拟器**（或调用 EPSS-Agent 的预演接口），在输出 $W^{\text{final}}(y)$ 后，立即在模拟环境中以该权重为输入、以快速仿真代替真实执行，预演若干典型方案的 VPD/OE 与耦合协调度，从而在不消耗真实年度的情况下预判权重合理性。当前设计中缺少这一" 数字孪生验证" 环节，**导致商分 Agent 与整个多 Agent 执行环境深度耦合**，丧失了独立迭代的能力。

2.1.9 商分 Agent 的进化阶段建议：黑箱工具 versus 自适应学习系统

前文表**??**将商分 Agent 封装为黑箱接口，强调" 调用者只需理解输入输出，无需掌握内部四层运算"。这一工程封装本身无可厚非，但它掩盖了一个根本性的定位困惑：**商分 Agent 究竟是一个即插即用的通用分析工具，还是一个需要在特定环境中持续学习收敛的专用智能体？**

若是前者（通用黑箱），则意味着先验分析层中的结构参数 $a_1 = 0.40, a_2 = 0.20, b_1 = 0.35, c_1 = 0.35$ 等、巡盘修正层中的映射系数 λ_i 以及融合系数 $\alpha = 0.65, \beta = 0.60$ **均具有跨环境的一般性。无论比赛规则如何调整、对手结构如何变化，只要输入详单与巡盘数据，Agent 即可输出" 合理" 权重**。这要求这些参数是经过大规模离线训练后固化的**通用知识，类似于预训练模型的权重**。

若是后者（专用学习系统），则意味着当面对一个新的商业环境（例如规则变更、新增市场、产品线扩展、对手策略范式迁移）时，商分 Agent 不能简单复用旧参数直接输出，而需要经历一个**环境适配期**：在新的沙盘实例中反复预测—预演—评估—修正，使 a_i, b_i, α, β 乃至反馈更新层的 μ, h_i, i_i, j_i 逐步收敛到与该环境匹配的局部最优。这更接近强化学习 Agent 的行为模式。

当前方案在表述上存在**定位漂移**，或者说商分 Agent 的定位需要课题组继续讨论：

1. 一方面，它强调" 首次调用时可使用均匀分布或专家给定值作为初始种子"（表**??**），暗示参数可随意初始化、无需预训练，偏向**自适应学习系统**的定位；
2. 另一方面，它又给出大量固定专家参数（$a_1 = 0.40, e_1 = 1.0, g_1 = 0.40$ 等），并在封装说明中强调" 可复用的 PSS 内部权重库"，暗示这些参数已是**固化知识**，偏向**通用黑箱工具**的定位。

这一矛盾在工程实践中将直接引发问题：若用户将商分 Agent 移植到一个全新的沙盘变体（例如产品数量从 4 种扩展到 8 种，或市场规则从” 四市场” 改为” 六市场”），原先由专家经验设定的 0.40/0.20/0.30/0.10 产品权重配比是否仍然成立？$\alpha = 0.65$ 的融合系数是否仍最优？方案未给出判断准则，也未提供**参数敏感性分析**或**超参数自动搜索**的机制。

改进建议：**课题组商分 Agent 团队应在 5 月的组会内，明确当前商分 Agent 的训练范式，并据此暴露不同的接口层级**，以下是不同工程难度、不同阶段，Agent 应该实现的功能与结构范式：

1. **Level-0（冻结模式）**：所有结构参数固化，适用于与训练环境完全一致的重复比赛，调用者将其视为黑箱；这样的设计只能用于初期验证，**在工程上想要全 Agent 闭环较快的落地成立**，此方案作为纯 Demo 是可行的。
2. **Level-1（微调模式）**：结构参数冻结，但反馈更新层开放 μ 与平滑窗口供调节，适用于环境小幅波动；此方案**类似迁移学习，具有一定的环境适应能力**，可以作为在工程 Demo 落地后，对 Agent 改进与进一步智能化的方向。
3. **Level-2（元训练模式）**：结构参数本身开放，支持基于历史沙盘库进行跨环境元学习（Meta-Learning），使 Agent 具备” **面对新规则时快速收敛**” 的能力；此方法可以放在 Level2 的参数训练机制成熟后，**对重要超参数进行参数化，使得环境适应能力更加强，现在的设计思路下，此方案是商分 Agent 的最终实现**。

唯有完成上述三级定位重新思考，商分 Agent 才能逐步地从一套” 公式化的年度权重计算器” 真正进化为可迁移、可验证、可自适应的商业环境认知智能体。

2.2 PSS-Agent 设计综述

在多 Agent 协作链路中，PSS-Agent 承担 “**经营单元建模—价值成本核算—协调性校验**” 的中间层职能。**商分 Agent 输出外部环境权重后，决策 Agent 会形成一组候选经营方案**；PSS-Agent 则把这些方案还原为可计算的 PSS 单元，逐一计算其 VPD/OE、生命周期状态、资产—效率—公平耦合水平，并把诊断结果回传给决策 Agent 与 EPSS-Agent。因此，PSS-Agent 不是单纯的财务汇总模块，而是用于判断 “某一产品、某一市场、某类生产线组合是否值得继续投入” 的细粒度经营评价智能体。

2.2.1 PSS 单元识别与当期数据提取

PSS-Agent 首先从当期 Excel 状态表中提取市场、产品、生产线、订单、库存、在建工程、研发、ISO、广告、采购、转产等数据，并形成标准化状态表。其最小建模单元定义为

$$PSS_i = (m_i, p_i, l_i), \quad m_i \in M,\ p_i \in P,\ l_i \in L \tag{24}$$

其中 m_i 表示市场，p_i 表示产品，l_i 表示生产线类型。PSS 数量由 “市场—产品—生产线类型” 的有效组合决定，而不是由同类生产线的物理条数决定。例如，若亚洲市场存在

两条自动线生产并销售 P1，该组合仍只对应一个“亚洲—P1—自动线”PSS；同类生产线的数量进入后续产量、机器时间、折旧维修费等分摊计算。

这种定义保留了市场差异、产品差异和产线差异三类决策信息。一方面，它可以支撑广告投放、订单选择和市场公平性分析；另一方面，它也能刻画转产、研发、产线利用率和机器成本分摊等内部经营问题。实际报告输出时，PSS-Agent 还会在全景清单基础上生成市场视角、产品视角、生产线视角，以及“产品—市场”“产品—生产线”等混景视角，便于不同 Agent 读取相同数据的不同切面。

2.2.2 PSS-Agent 当期计算流程

PSS-Agent 的当期运行流程可概括为“提数—计算—更新—校验—输出”。其核心流程如图6所示。

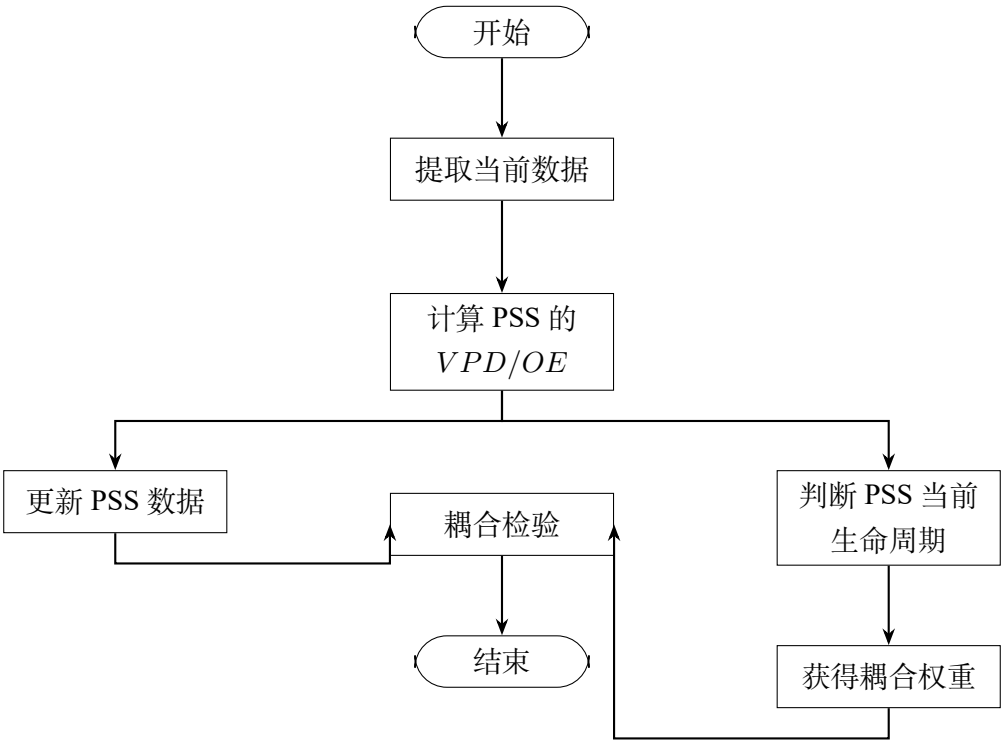

图 6: PSS-Agent 当期计算流程图

其中，提取当前数据阶段负责把原始 Excel 表格转换为 PSS 级状态表；计算阶段根据 PSS 组合计算 VPD、OE 及其比值；更新阶段将计算结果回写至 PSS 数据表，形成跨年度、跨季度的历史序列；生命周期判断阶段依据当期和累计 VPD/OE 识别该 PSS 所处阶段；耦合权重获取阶段调用已训练或设定的权重函数；耦合检验阶段判断当前方案是否在资产、效率、公平三个维度上同时达到可接受水平。

2.2.3 VPD/OE 核算与费用分摊规则

PSS-Agent 以 VPD/OE 作为单个 PSS 的核心健康度指标：

$$R_i = \frac{VPD_i}{OE_i} \tag{25}$$

其中 VPD_i 刻画该 PSS 在当期创造的价值，主要包括本期该产品在当前市场的净利润、对应原材料采购成本以及原材料运输成本；OE_i 刻画该 PSS 占用的运营资源，主要包括机器折旧费、机器维修费、产品加工费、转产费用、转产调整费、研发费用、ISO 费用、市场开拓费用及其他需要归集的综合费用。

费用分摊遵循“谁受益、谁承担；谁占用、谁分摊”的原则。若同一 PSS 对应多条同类生产线，则先提取这些生产线的全部机器数据，再按照机器可用时间、产量、销售去向或计划产线期数进行分摊。若生产线发生转产，同一条生产线在一年内可拆分为多个机器数据片段，维修费、折旧费按可用季度等比例分配，转产费用和转产调整费由转入后的 PSS 承担。若存在市场销售分布差异，则同一生产线生产出的产品还需按照不同市场的销售数量或计划产量比例拆分到对应市场 PSS。

对于具有延时效应的投入，PSS-Agent 采用累计确认机制。若生产线当期尚未建设完成，则该 PSS 当期 $VPD = 0$，但已发生的建设投入仍进入 OE 并累计到后续期间；下一次完成计算时，OE 需要叠加此前未形成产出的投入。研发费用、ISO 认证费用和市场开拓费用同样遵循这一逻辑：未完成时先累计，完成并产生经营效应后再进入相应 PSS 的完整核算。

2.2.4 生命周期、耦合权重与方案校验

在完成 VPD/OE 计算后，PSS-Agent 会同时记录当期比值与累计比值，并据此判断 PSS 所处阶段。不同阶段的 PSS 对资产增长、运营效率和公平性的敏感度不同，因此系统不采用固定的人为权重，而是通过权重函数

$$F_{\text{Weight}}\left(P_{VPD/OE}, Type_U\right) \tag{26}$$

根据 PSS 阶段和指标类型动态生成子指标权重。其中 $Type_U$ 对应资产、效率、公平等指标类型，$P_{VPD/OE}$ 表示由当期及累计 VPD/OE 共同刻画的生命周期状态。

耦合校验围绕三类指标展开：资产指标 U_1 衡量 PSS 资产及其增速，效率指标 U_2 衡量内部产线使用与外部行业对比，公平指标 U_3 衡量市场公平与供应商公平。三个指标经标准化后，先进行两两耦合分析，得到 H_{12}、H_{13}、H_{23}，再进行三指标耦合分析，得到整体协调度 H。只有当整体协调度和局部协调度均处于可接受范围时，当前 PSS 方案才会被确认；若某一局部协调度显著偏低，PSS-Agent 会把问题定位到资产—效率、资产—公平或效率—公平的具体矛盾，并将诊断结果返回给决策 Agent 重新调整。

2.2.5 PSS-Agent 输出报告

PSS-Agent 最终输出面向开发与决策复盘的结构化报告。报告至少包含 PSS 全景清单、单景清单、混景清单、颗粒度成本明细、指标标准化结果、两两耦合结果、三指标耦合结果、诊断意见与下期优化方向。对于 PSS 数量较多的情况，报告可选择销售额最高、VPD/OE 波动最大或协调度最低的代表性 PSS 绘制折线图，同时展示当期 VPD/OE 与累计 VPD/OE，用于观察该 PSS 是否处于导入、成长、成熟或衰退状态。

从工程封装角度看，PSS-Agent 向外暴露的是一个稳定的黑箱接口：输入为当期状态表、规则参数、历史 PSS 数据和候选决策方案；输出为 PSS 数据更新表、VPD/OE 结果、耦合校验结果和诊断报告。内部的成本拆分、生命周期判断与权重学习逻辑可以持续迭代，但对上游决策 Agent 和下游 EPSS-Agent 而言，只需要遵循该输入输出契约即可完成联调。

2.3 EPSS–Agent 智能体设计综述

EPSS-Agent（Enterprise-level PSS Agent）承担企业全局层面的指标聚合与综合诊断职能。它在年度、季度、年末三个时间粒度上，接收 PSS-Agent 输出的细粒度经营单元数据，按” 产品—生产线” 二维折叠聚合，计算企业级 VPD/OE、竞争系数与耦合协调度，并将诊断结果回传至商分 Agent 与决策 Agent，形成” 单元核算—全局聚合—策略反馈” 的闭环。

2.3.1 年初：聚合 PSS 与竞争格局预判

（1）PSS 层到 EPSS 层的输入映射

前序 PSS-Agent 将企业内部单元按” 产品—生产线—市场” 三维识别。设第 k 个原始 PSS 单元为：

$$PSS_{\text{原始}}^{(k)} = \left(p^{(k)}, l^{(k)}, m^{(k)}\right)$$

其中 $p^{(k)}$、$l^{(k)}$、$m^{(k)}$ 分别为该单元对应的产品类型、生产线类型与市场类型。

年度决策层并不直接使用三维原始 PSS，而是将市场维度折叠，仅保留” 产品—生产线” 二维作为年度总方案中的决策单元。设第 i 个聚合后 PSS 单元为：

$$PSS_i = (p_i, l_i)$$

（2）聚合后 PSS 的参数计算（年初整年视角）

年度价值密度 VPD 与运营成本 OE 按以下规则核算：

VPD 核算：

$$\text{VPD}_i = \text{净利润}_i + \text{贷款利息均摊}_i + \text{原材料采购成本}_i + \text{原材料运输成本}_i$$

其中：

1. 贷款利息均摊 = 当年全部贷款利息 / 聚合后 PSS 总数；
2. 原材料采购成本 = 产品数量 × 每个产品所需原材料个数 × 原材料单价；
3. 原材料运输成本：年初视角下取 0（运输成本在季度执行中计入）。

OE 核算：

$$\begin{aligned}\text{OE}_i = {} & \text{当年产线建设费}_i + \text{当年折旧费与维修费}_i + \text{其他费用均摊}_i \\ & + \text{当年研发投入}_i + \text{产品当年加工费}_i + \text{转产费}_i + \text{转产调整费}_i \end{aligned} \tag{27}$$

其中：

1. 当年产线建设费 = 本年此 PSS 对应产线的建设费（总建设费 − 历史已投入建设费）；
2. 机器折旧费与维修费：按本年在当前 PSS 使用的机器分摊，如有转产则按使用时间比例分配；若有产线建设费则不计算折旧；
3. 当年研发成本 = 产品总研发成本 − 已投入研发成本；
4. 产品加工费统一为每件 10 万元，按当前产品、当前生产线求和。

年度价值密度：

$$\rho_i^{\text{year}} = \frac{\text{VPD}_i}{\text{OE}_i}$$

（3）竞争系数（按聚合后 PSS 计算）

竞争系数刻画不同 PSS 单元之间的资源争夺关系，构建三维竞争系数向量：

$$\vec{C}_{ij} = \left(C_{ij}^{\text{物料}}, C_{ij}^{\text{产线}}, C_{ij}^{\text{市场}}\right)$$

物料竞争系数：基于两个 PSS 单元对同类原材料的需求金额重叠度计算。设 PSS_i 与 PSS_j 的共用物料集合为 $S_{ij} = M_i \cap M_j$。若 $S_{ij} = \emptyset$，则 $C_{ij}^{\text{物料}} = 0$；否则：

$$C_{ij}^{\text{物料}} = \sum_{s \in S_{ij}} w_s \cdot \frac{\min(D_{is}, D_{js})}{\max(D_{is}, D_{js})}$$

其中 D_{is} 为物料 s 在 PSS_i 中的预计需求金额，w_s 为物料重要性权重（按订购金额和提前期判断）。

产线竞争系数：针对同一生产线类型上的产品转换关系计算。若 $p_i \neq p_j$ 且 $l_i = l_j$，即两 PSS 共用同类型产线但生产不同产品，则需考虑转产的机会成本。从商分 Agent 获取本年度产品权重 w_{p_i}, w_{p_j}，当 $w_{p_j} > w_{p_i}$ 时，评估从 p_i 向 p_j 转产的可行性。转产所用季度数与历史生产季度数 n（本年转产前该类型生产线累积生产 p_i 的季度总和）及规则中转产时间 T_{conv} 相关。

订单交单竞争系数：基于订单交单情况的竞争关系，后续与现金流工具对接实现动态更新。

2.3.2 季度：执行中 PSS 的动态监控

季度层面的计算与年初逻辑一致，但需注意两项特殊处理：

（1）建设未完成与延时效应

若生产线当期尚未建设完成，则该 PSS 当期 VPD $= 0$，但已发生的建设投入仍进入 OE 并累计到后续期间。下一次完成计算时，OE 需要叠加此前未形成产出的投入。研发费用、ISO 认证费用和市场开拓费用同理：未完成时先累计，完成并产生经营效应后再进入相应 PSS 的完整核算。

（2）季度参数核算

季度 VPD：

$$\text{VPD}_i^{\text{quarter}} = \text{本期净利润}_i + \text{本期贷款利息均摊}_i + \text{本期原材料采购成本}_i + \text{本期原材料运输成本}_i$$

季度 OE 在年初基础上增加厂房租赁成本与管理费：

$$\begin{aligned}\text{OE}_i^{\text{quarter}} = {} & \text{当期折旧维修或建设费} + \text{ISO 均摊} + \text{研发均摊} + \text{当期加工费} \\ & + \text{转产费} + \text{转产调整费} + \text{厂房租赁成本}_i + \text{管理费}_i\end{aligned} \tag{28}$$

其中：

1. 厂房租赁成本：按厂房状态确定总成本，依据该厂房内该类在线生产线数量与占用时间分摊到各 PSS；
2. 管理费：若与厂房数无关，则每季收取基本管理费；若与厂房数有关，则按厂房数量阶梯计费（0–2 座为基本管理费，3–4 座为基本管理费 $\times 2$），再按与厂房成本相同的规则分摊到 PSS。

季度价值密度：

$$\rho_i^{\text{quarter}} = \frac{\text{VPD}_i^{\text{quarter}}}{\text{OE}_i^{\text{quarter}}}$$

若 $\rho_i^{\text{quarter}} < 1$，说明该产品—产线组合本季度投入产出效率较低，需触发预警。

季度竞争系数沿用年初判断的物料关系，产线竞争在季度层面按规则不会出现（转产决策已在年初确定），订单交单竞争系数后续与现金流工具对接。

2.3.3 年末：全局聚合与竞争系数汇总

年末对全年数据进行最终汇总。

（1）年度 VPD 与 OE 汇总

$$\text{VPD}_i^{\text{year-end}} = \sum_{\text{quarters}} \text{VPD}_i^{\text{quarter}}, \quad \text{OE}_i^{\text{year-end}} = \sum_{\text{quarters}} \text{OE}_i^{\text{quarter}}$$

（2）年末竞争系数汇总

物料竞争：若存在共用物料，先计算年度物料需求金额

$$D_{is}^{\text{year}} = Q_i^{\text{year}} \cdot q_{p_i,s} \cdot P_s$$

其中 Q_i^{year} 为 PSS_i 年度产量，$q_{p_i,s}$ 为生产一件产品 p_i 需要物料 s 的数量，P_s 为物料 s 单价。

年度物料竞争系数：

$$C_{ij}^{\text{物料,year}} = \sum_{s \in S_{ij}} w_s \cdot \frac{\min(D_{is}^{\text{year}}, D_{js}^{\text{year}})}{A_s^{\text{supply}}}$$

其中 A_s^{supply} 为物料 s 的年度供应能力金额（当年采购金额）。

产线竞争：若本年度发生转产，则按年初计算规则执行；未发生转产则无需重复计算。

2.3.4 EPSS 企业级指标计算

每个企业仅有一个 EPSS。在聚合后的各 PSS 年末数据基础上，进行企业级汇总：

$$\text{VPD}_{\text{EPSS}} = \sum_i \text{VPD}_i^{\text{year-end}}, \quad \text{OE}_{\text{EPSS}} = \sum_i \text{OE}_i^{\text{year-end}}$$

企业级价值运营比：

$$R_{\text{EPSS}} = \frac{\text{VPD}_{\text{EPSS}}}{\text{OE}_{\text{EPSS}}}$$

该比值用于判定企业本年度整体所处生命周期（导入期、成长期、成熟期、衰退期），并将此数据返回商分 Agent，作为下一年度权重调整的重要依据。

2.3.5 耦合协调度评估：资产—效率—公平三维诊断

EPSS-Agent 在年初、季度、年末三个时点均需执行耦合协调度分析，以诊断决策方案的整体协调性与可持续性。

（1）三类核心指标

资产指标 U_1：衡量企业总资产及其增速。

$$U_1 = \mu \cdot (U_{1.1} \times U_{1.2})$$

其中 $U_{1.1}$ 为企业总资产（现金 + 应收款 + 在制品 + 产成品 + 原料 + 流动资产 + 厂房 + 机器设备 + 在建工程 − 贷款总额 − 所得税），$U_{1.2}$ 为公司资产增速 = (期末总资产 − 期初总资产)/ 期初总资产。

效率指标 U_2：衡量企业发展潜力，包含内部效率与外部效率。

$$U_2 = \alpha \cdot U_{2.1} + (1-\alpha) \cdot U_{2.2}$$

其中 $U_{2.1}$ 为企业综合发展潜力 /100（市场资格分值 +ISO 资格分值 + 生产资格分值 + 厂房分值 + 各生产线分值求和）；$U_{2.2}$ 为本企业综合发展潜力 / 本组企业综合发展潜力平均值。生产线建成（含转产）即加分，厂房须为购买状态。

公平指标 U_3：衡量市场公平与供应商公平。

$$U_3 = \beta \cdot U_{3.1} + (1-\beta) \cdot U_{3.2}$$

其中 $U_{3.1}$ 为本企业内每个 PSS 市场公平 / 行业平均市场公平的平均值，市场公平 = 销售额 / 广告投入额（无广告投入时取 1）；$U_{3.2}$ 为本企业内每个 PSS 供应商公平 / 行业平均供应商公平的平均值，供应商公平 = 该 PSS 对应生产线已卖出产品的原料成本 / 该 PSS 对应生产线当季线上在制品与当年起至该期已出产产品的总原料成本。

上述指标经标准化到 $[0,1]$ 区间后进入耦合分析，α, β, μ 为协调系数，默认取 0.5，留作与其他智能体对接的调参接口。

（2）两两耦合分析

用于发现局部不平衡。以资产—效率耦合为例：

$$C_{12} = \sqrt{\frac{U_1 \cdot U_2}{(U_1+U_2)/2}}, \quad F_{12} = \beta_1 U_1 + \beta_2 U_2, \quad H_{12} = \sqrt{C_{12} \cdot F_{12}}$$

其中 C_{12} 为耦合度，F_{12} 为综合发展得分，H_{12} 为耦合协调度。资产—公平耦合 H_{13}、效率—公平耦合 H_{23} 同理计算。

（3）三指标耦合分析

用于评估决策的整体协调性：

$$C = \sqrt[3]{\frac{U_1 \cdot U_2 \cdot U_3}{(U_1+U_2+U_3)/3}}, \quad F = \beta_1 U_1 + \beta_2 U_2 + \beta_3 U_3, \quad H = \sqrt{C \cdot F}$$

（4）综合诊断标准

依据总协调度 H 进行整体诊断：

1. $H > 0.8$：高度协调，优质决策；
2. $0.5 < H \leq 0.8$：中度协调，决策基本合理；
3. $H \leq 0.5$：低度协调或失调，决策存在严重问题。

同时查看两两协调度 H_{12}, H_{13}, H_{23} 进行局部诊断，定位资产—效率、资产—公平或效率—公平的具体矛盾，并将诊断结果返回决策 Agent 重新调整。

三　沙盘重构上的多 Agent 执行与计算平台

前文已完成了商分 Agent、PSS-Agent 与 EPSS-Agent 的算法设计与改进方向论证。然而，多 Agent 协同决策系统的真正落地，不仅需要单个 Agent 的公式正确性，更需要一个**可执行**、**可观测**、**可对接**的沙盘运行平台。课题组选择以百树电子沙盘为底层规则基座，对其进行面向智能体决策的系统性重构，构建一套支持全自动对抗、人机混合对抗与远程可视监控的多 Agent 执行与计算平台。

3.1　百树沙盘平台规则综述

百树沙盘是一套基于 ERP 思想的商业模拟系统，其规则可抽象为**资源约束下的多周期动态博弈**。课题组在保留百树原有规则内核的前提下，将其全部规则参数化、引擎化，使 Agent 能够直接读取规则表而非依赖人工解释。百树沙盘的核心规则模块包括：

（1）生产线规则。百树沙盘提供四类生产线，其技术经济参数如表3所示。

表 3: 百树沙盘生产线技术经济参数

线型	投资总额	安装周期	生产周期	转产费	转产周期	维护费	残值	折旧费	分值
手工线	50W	0 季	3 季	0W	0 季	10W/年	10W	10W	5
半自动	100W	1 季	2 季	10W	1 季	10W/年	20W	20W	7
自动线	150W	3 季	1 季	20W	1 季	10W/年	30W	30W	9
柔性线	200W	4 季	1 季	0W	0 季	10W/年	40W	40W	10

关键约束包括：安装周期为 0 表示即买即用；计算投资总额时若安装周期为 0 则按 1 折算；不论何时出售生产线价格均为残值，净值与残值之差计入损失；只有空生产线方可转产；当年建成生产线需交维护费；采用平均年限法折旧，建成当年不提折旧，净值等于残值时停止计提。

（2）融资规则。融资体系包含长期贷款、短期贷款与资金贴现三类工具。长贷每年年初发放，年息 10%，年初付息到期还本，额度受上年权益 3 倍约束；短贷每季度初发放，年息 5%，到期一次还本付息，同样受权益倍数约束；资金贴现可在任何时间进行，1–2 期应收款贴现率 10%，3–4 期贴现率 12.5%。此外，库存拍卖按产品 100%、原料 80% 折价变现。

（3）厂房与市场规则。厂房分为大厂房（300W，容量 4 条线，分值 10）与小厂房（200W，容量 3 条线，分值 7），可选择购买或租赁。厂房出售得到 4 个账期的应收款，紧急情况下可厂房贴现直接获得现金。市场开拓方面，本地、区域、国内、亚洲、国际五个市场分别需要 1–4 年开发时间，每年开发费 10W，开发完成后领取资格证。ISO 认证包含 ISO9000（2 年，每年 10W，分值 8）与 ISO14000（2 年，每年 20W，分值 10）。

（4）产品研发与原料规则。四种产品 P1–P4 的加工费均为 10W，直接成本分别为 20W、30W、40W、50W，研发周期 2–5 季不等。原料 R1–R4 购买单价均为 10W，提前

期 1–2 季。原料紧急采购价格为直接成本的 2 倍，成品紧急采购价格为直接成本的 3 倍。

（5）经营与评分规则。选单规则依次考虑市场老大（上年本市场销售额最高且无违约者优先）、本市场本产品广告额、本市场广告总额、市场销售排名。破产标准为现金断流或权益为负。最终总成绩计算公式为：

$$\text{总成绩} = \text{所有者权益} \times \left(1 + \frac{\text{企业综合发展潜力}}{100}\right)$$

其中企业综合发展潜力 = 市场资格分值 + ISO 资格分值 + 生产资格分值 + 厂房分值 + 各条生产线分值。

上述规则的全部参数被抽取为**标准化规则表**（JSON/YAML 格式），供 Agent 直接读取。规则引擎的核心设计思想是：任何规则变更（如调整贴现率、修改生产线参数）只需更新规则表，无需修改 Agent 代码，实现规则与策略的解耦。

3.2 平台运行的基本控制流思路

平台的控制流设计遵循状¨态驱动、事件触发、Agent 决策、状态更新的¨闭环逻辑。整个比赛被抽象为一个**离散时间状态机**，时间粒度为年¨度–季度，¨每个时点上系统维护一份完整的企业状态快照，Agent 在此基础上进行决策，决策结果经规则引擎验证后更新状态快照。

3.2.1 年度级控制流

一个标准年度（第 1–6 年）的控制流如下：

1. **年初状态初始化**：系统读取上年末状态，初始化本年度年初快照，包括现金、应收款、在制品、产成品、原料、生产线、厂房、贷款、资格认证等全部状态变量。
2. **商分 Agent 调用**：商分 Agent 读取历史详单、规则表与竞争对手巡盘数据（ProductLine/AD），输出本年度市场权重、产品权重、产线权重及组合优先级。此步骤仅在每年第一季度开始前执行一次。
3. **长贷与短贷决策**：决策 Agent 根据现金流预测与商分权重，生成长贷申请额与全年短贷计划。长贷在年初一次性发放，短贷计划在每季度初按需触发。
4. **广告投放决策**：选单 Agent 根据产品–市场优先级矩阵 $\{S_{pm}\}$，确定各产品–市场组合的广告投放额，生成 AD 矩阵。
5. **选单与竞单**：系统按百树选单规则排序，Agent 依次选择订单。竞单环节（如有）单独处理，时间窗口 90 秒，同竞数 3。
6. **四季度循环**：每季度内部按季¨初短贷 → 原料采购 → 生产排程 → 产品入库 → 交货收款 → 季末费用结算顺¨序执行。Agent 在每个决策点被调用，输入当前状态快照，输出决策动作（采购量、生产计划、交货选择等）。
7. **年末结算**：执行折旧、维修费、管理费、市场/ISO/研发投资分摊、所得税计算、权益更新。PSS-Agent 与 EPSS-Agent 被调用，计算各 PSS 单元的 VPD/OE 与耦合协

调度 H。

8. **反馈更新**：商分 Agent 读取本年度实际经营结果（中标率、利润率、VPD/OE、耦合协调度），更新下一年度权重种子。

3.2.2 状态机与数据流设计

平台内部维护三类核心数据结构：

1. **全局状态表**（Global State Table, GST）：记录所有参赛企业的完整经营状态，包括资产负债表、利润表、现金流量表的三表联动数据。GST 在每年初从文件系统加载，每季度末持久化回存。
2. **Agent 决策缓存**（Agent Decision Cache, ADC）：记录每个 Agent 在每个决策时点的输入状态与输出动作，形成可追溯的决策链。ADC 是后续复盘、训练与策略优化的核心数据资产。
3. **事件日志**（Event Log, EL）：记录所有规则触发事件（如选单成功、违约发生、破产判定、转产完成等），支持按时间线回放与异常审计。

控制流引擎采用**生产者–消费者模式**：规则引擎作为生产者，按时间顺序生成决策事件；Agent 调度器作为消费者，根据事件类型匹配对应的 Agent 实例，调用其决策接口，并将返回的决策动作回传给规则引擎执行验证。若决策动作违反规则约束（如现金流不足、资格未获得、产能超限），规则引擎返回错误码，Agent 需在限定时间内重试或进入默认处理分支。

3.3 平台表现形式：CLI 与基于 web 的 GUI

平台面向两类用户群体设计差异化的交互界面：**CLI（命令行界面）**面向开发者与算法调试人员，强调快速迭代、批量测试与可脚本化；**Web GUI** 面向比赛组织者、观摩者与策略分析师，强调可视化、实时监控与历史回放。

3.3.1 CLI 表现形式

CLI 是平台的核心执行入口，其设计目标是实现一¨条命令启动整场比赛。¨主要功能模块包括：

1. **比赛初始化**：`python platform.py init --rules rules.json --teams 10 --years 6`，读取规则表与参赛队伍配置，生成初始状态快照。
2. **Agent 注册**：`python platform.py register --agent shangfen --module agents/shangfen.py`，将各 Agent 模块动态加载到调度器中。Agent 以 Python 模块形式接入，遵循统一的 `BaseAgent` 抽象类。
3. **单步执行**：`python platform.py step`，执行下一个决策事件并打印状态变更。适用于 Agent 调试与策略单步追踪。

4. **批量运行**：`python platform.py run --batch 100`，批量运行 100 场比赛，输出统计数据（平均权益、胜率分布、策略收敛曲线）。适用于超参数搜索与策略对比。
5. **日志回放**：`python platform.py replay --log 2026-05-23-match-01.el`，按时间线回放某场比赛的全部决策过程，支持断点暂停与状态探查。

CLI 的输出采用结构化格式（JSON Lines），便于与外部数据分析工具（Pandas、Jupyter Notebook）对接。所有状态变更、Agent 决策、规则触发事件均以 JSON 对象形式输出到 stdout 或指定日志文件。

3.3.2 基于 Web 的 GUI 表现形式

Web GUI 面向远程观摩与实时监控场景设计，采用**前后端分离**架构。前端基于 React/Vue 框架实现响应式仪表盘，后端提供 RESTful API 与 WebSocket 实时推送。

网络架构：课题组现有计算服务器部署平台核心引擎（规则引擎、Agent 调度器、状态机），承担全部计算负载。计算服务器通过 frp（Fast Reverse Proxy）与阿里云公网服务器（120.26.111.75，已完成 ICP 备案）建立反向隧道，将后端 API 端口映射到公网域名。外部用户通过浏览器访问阿里云服务器即可实时观摩比赛，无需直连课题组内网。

该架构的优势在于：

1. 计算密集型任务（Agent 推理、状态机更新、耦合协调度计算）全部在课题组本地高性能服务器上执行，充分利用现有硬件资源；
2. 阿里云服务器仅承担流量转发与静态资源分发，负载极低，一台 1 核 2G 配置即可支撑百人同时在线观摩；
3. frp 隧道支持 TLS 加密与心跳保活，公网访问安全性与稳定性可控；
4. 备案后的域名可直接用于学术展示、比赛直播与远程评审，无需担心网络合规问题。

Web GUI 核心功能模块：

1. **实时对战大屏**：以沙盘地图形式展示各参赛企业的生产线布局、广告投放热力图、订单分布与现金流动态。大屏刷新频率与平台内部时钟同步，延迟控制在 1 秒以内。
2. **Agent 决策透视**：点击任意企业，可展开该企业各 Agent 的决策链视图，包括商分 Agent 输出的权重向量、决策 Agent 生成的候选方案、选单 Agent 的投标序列等。该功能对策略研究与教学复盘至关重要。
3. **PSS 健康度监控**：以雷达图形式实时展示各 PSS 单元的 VPD/OE、生命周期状态与竞争系数。当某 PSS 价值密度低于阈值时，系统自动标红并推送预警。
4. **耦合协调度仪表盘**：以动态折线图展示 U_1（资产）、U_2（效率）、U_3（公平）三类指标的时序变化，以及两两耦合协调度 H_{12}, H_{13}, H_{23} 与总协调度 H 的走势。
5. **历史回放与对比分析**：支持按时间轴拖拽回放任意已完成比赛，并可将两场比赛的同一指标进行叠加对比（如本 Agent 今年与去年的权益曲线对比，或本方与冠军队伍的耦合协调度对比）。

3.4 现有设计 Agent 的接入协议规范

为实现不同 Agent 模块的即插即用与跨团队复用，课题组制定了**三层接入协议**：接口抽象层、通信协议层与数据格式层。

3.4.1 接口抽象层与 ERPAI 协议范式

课题组将多 Agent 系统的接口规范统一命名为 **ERPAI 协议**（ERP Agent Interface Protocol），旨在建立一套面向 ERP 沙盘智能体的标准化接入范式。ERPAI 协议对标 OpenAI API 的设计理念，强调**输入输出标准化、调用方式统一化、状态持久化规范化**，使不同功能定位的 Agent 能够在同一平台上即插即用、协同演化。

所有 Agent 必须继承自统一的 `BaseAgent` 抽象基类，并实现以下核心方法：

1. `setup(config)`：在比赛初始化时调用，传入 Agent 专属配置参数（如学习率 μ、融合系数 α, β、结构参数 a_i, b_i 等）。
2. `decide(state_snapshot)`：在每个决策时点被调度器调用，输入当前状态快照，输出决策动作字典。该方法必须在规定超时时间内返回（默认 5 秒），否则触发默认策略。
3. `feedback(result_dict)`：在年末或季度末被调用，传入实际经营结果，供 Agent 进行内部权重更新与策略修正。
4. `teardown()`：在比赛结束或 Agent 被注销时调用，释放资源并持久化内部状态。

在此基础上，ERPAI 协议对三类核心 Agent 的输入输出进行范式化封装，如表4所示。

表 4: ERPAI 协议：Agent 黑箱封装接口规范范式

接口项	商分 Agent	PSS-Agent	EPSS-Agent
功能定位	年度初将历史数据、规则与巡盘信息转化为可执行权重与优先级	经营单元建模、价值成本核算、协调性校验	企业级 PSS 聚合、综合诊断、策略反馈
必需输入	1. 历史详单（.xlsx）；2. 静态规则（.xlsx）；3. 年初巡盘数据（ProductLine/AD）	当期 Excel 状态表：市场、产品、生产线、订单、库存、在建工程、研发、ISO、广告、采购、转产等	各 PSS 单元的 VPD/OE、竞争系数向量、企业级状态数据（资产负债表、利润表、现金流量表）
可选输入	上年度反馈权重 $w(y)$（首次可用均匀分布或专家值作为种子）	历史 PSS 数据、商分 Agent 输出的环境权重	历史 EPSS 数据、行业平均市场公平与供应商公平基准
输出参数	五类权重/优先级向量：$\{w_p\},\{w_m\},\{w_l\},\{S_{pm}\},\{S_{pl}\}$，均经 Softmax 归一化	PSS 全景/单景/混景清单、VPD/OE 结果、生命周期判定、耦合校验结果、诊断报告	EPSS 企业级 VPD/OE、生命周期判定、耦合协调度 H、两两耦合 $H_{12}/H_{13}/H_{23}$、诊断意见与优化方向
调用时机	每年度第一季度开始前执行一次；重大变故可触发额外调用	每季度末执行一次；年末执行完整年度核算与反馈	年初/季度末/年末三个时点均执行，与 PSS-Agent 联动
依赖关系	上游：详单解析、规则读取；下游：决策 Agent、选单 Agent	上游：商分 Agent、决策 Agent；下游：EPSS-Agent、决策 Agent	上游：PSS-Agent；下游：商分 Agent、决策 Agent
状态持久化	年度权重更新表（旧权重、建议新权重、平滑后权重、归一化权重）	PSS 数据更新表、VPD/OE 历史序列、跨季度累计 OE 台账	EPSS 年度聚合表、耦合协调度历史序列、诊断报告库

3.4.2 以商分 Agent 运行为例的接口运行与数据接入流程

为具象化 ERPAI 协议在实际比赛中的运行方式，以下以**第 3 年度年初商分 Agent 的完整决策链路**为例，展示从**数据获取**、**Agent 计算**、**结果注入平台到反馈闭环的全流程**。

参数均取自百树沙盘标准规则与课题组实测数据。

步骤 1：平台发起调用并传递状态快照。第 3 年年初，平台规则引擎触发 `shangfen_agent.decide(state_snapshot)`，传入的状态快照核心字段如下（JSON 片段）：

```
{
  "meta": {"year": 3, "quarter": 1, "team_id": "ZC01"},
  "rules": {
    "products": ["P1","P2","P3","P4"],
    "markets": ["本地","区域","国内","亚洲","国际"],
    "lines": ["手工线","半自动","自动线","柔性线"],
    "material_cost": {"P1":20, "P2":30, "P3":40, "P4":50}
  },
  "historical_detail": [
    {"year":1, "product":"P2", "market":"本地", "qty":120, "price":95},
    {"year":1, "product":"P3", "market":"国内", "qty":80, "price":110},
    {"year":2, "product":"P2", "market":"本地", "qty":150, "price":98},
    {"year":2, "product":"P3", "market":"国内", "qty":110, "price":115}
  ],
  "productline_inspect": {
    "P2_active_teams": 9,
    "P3_active_teams": 7,
    "auto_lines_for_P2": 12,
    "flex_lines_for_P3": 5
  },
  "ad_inspect": {
    "P2-本地": {"total_ad": 180, "max_rival_ad": 45},
    "P3-国内": {"total_ad": 120, "max_rival_ad": 60}
  },
  "feedback_seed": {
    "w_product": {"P1":0.15, "P2":0.35, "P3":0.30, "P4":0.20},
    "w_market": {"本地":0.25, "区域":0.20, "国内":0.30, "亚洲":0.15, "国际":0.10}
  }
}
```

步骤 2：商分 Agent 内部运算。Agent 按四层架构执行：

1. **先验分析层**：基于历史详单计算 GroupMeanQty$(P2,3)=135$，GroupMeanQty$(P3,3)=95$，得需求强度 $D_{P2}=0.72$，$D_{P3}=0.58$；增长趋势 $G_{P2}=+0.25$，$G_{P3}=+0.375$；价格强度 $P_{P2}=98$，$P_{P3}=115$。代入先验公式得 $S_{P2}^{\text{prior}}=0.68$，$S_{P3}^{\text{prior}}=0.59$。

2. **巡盘修正层**：ProductLine 显示 P2 有 9 队竞争、自动线 12 条，供给压力 $\text{SupplyPress}_{P2} = 0.85$，产品修正量 $\Delta S_{P2}^{\text{PL}} = -0.85$；AD 显示 P3-国内市场广告集中度 ADConc $= 0.50$，修正量 $\Delta S_{P3-\text{国内}}^{\text{AD}} = -0.15$。
3. **权重融合层**：融合后 $S_{P2}^{\text{final}} = 0.68 - 0.85 = -0.17$，$S_{P3}^{\text{final}} = 0.59 - 0.15 = 0.44$。经 Softmax 归一化得产品权重 $w_{P2}^{\text{final}} = 0.28$，$w_{P3}^{\text{final}} = 0.42$。
4. **反馈更新层暂不参与**：反馈更新在年末执行，年初阶段仅读取 `feedback_seed` 中的上年权重作为平滑初始值。

步骤 3：Agent 返回决策动作。`decide()` 方法返回 JSON 格式的决策动作字典：

```
{
  "agent_type": "shangfen",
  "action_type": "WEIGHT_OUTPUT",
  "output": {
    "w_product": {"P1":0.12, "P2":0.28, "P3":0.42, "P4":0.18},
    "w_market": {"本地":0.22, "区域":0.18, "国内":0.35,
                 "亚洲":0.15, "国际":0.10},
    "w_line": {"手工线":0.10, "半自动":0.20,
               "自动线":0.45, "柔性线":0.25},
    "S_pm": {"P3-国内":0.85, "P2-本地":0.60,
             "P4-亚洲":0.45, "P1-本地":0.30}
  },
  "confidence": 0.78,
  "execution_timestamp": "2026-05-23T09:00:00+08:00"
}
```

步骤 4：平台接收权重并驱动下游决策。平台规则引擎将商分 Agent 输出的权重注入全局状态表 GST，随后触发下游 Agent 调用：

1. **决策 Agent** 读取 $w_{P3} = 0.42$ 与 $S_{P3-\text{国内}} = 0.85$，判定”P3-国内” 为本年度核心突破口，生成三套产能配置方案（保守/均衡/激进），其中激进方案建议新建 2 条自动线专产 P3、追加国内 ISO14000 认证投入。
2. **选单 Agent** 根据 $w_{\text{国内}} = 0.35$ 与 $S_{P3-\text{国内}} = 0.85$，在国内市场 P3 广告位投放年度预算的 40%（假设年度广告预算 200W，则 P3-国内分配 80W），其余市场按权重递减分配。

步骤 5：年度执行与年末反馈。第 3 年四季度全部执行完毕后，平台汇总实际经营结果：P3-国内实际中标率 WinRate $= 65\%$，实际利润率 ProfitRate $= 22\%$，对应 PSS 单元 VPD/OE$= 1.35$，耦合协调度 $H = 0.72$。平台将上述结果封装为 `result_dict`，调用 `shangfen_agent.feedback(result_dict)`。

步骤 6：反馈更新与下一次循环。商分 Agent 内部执行反馈更新层：

1. 计算建议新权重：$\hat{w}_{P3}(4) = 0.42 + 0.35 \times 0.65 + 0.40 \times 0.22 = 0.72$；
2. 平滑更新（$\mu = 0.15$）：$w_{P3}(4) = 0.85 \times 0.42 + 0.15 \times 0.72 = 0.465$；
3. 归一化后写入年度权重更新表，作为第 4 年年初 `feedback_seed` 的初始种子。

至此，商分 Agent 完成了一次完整的” 预测—执行—评估—再学习” 循环。该循环在 ERPAI 协议的框架下被标准化为六个确定性步骤，任何符合 ERPAI 接口规范的 Agent 均可按同一模式接入平台，无需关心底层规则引擎的实现细节。上述流程可用图**??**直观表示。

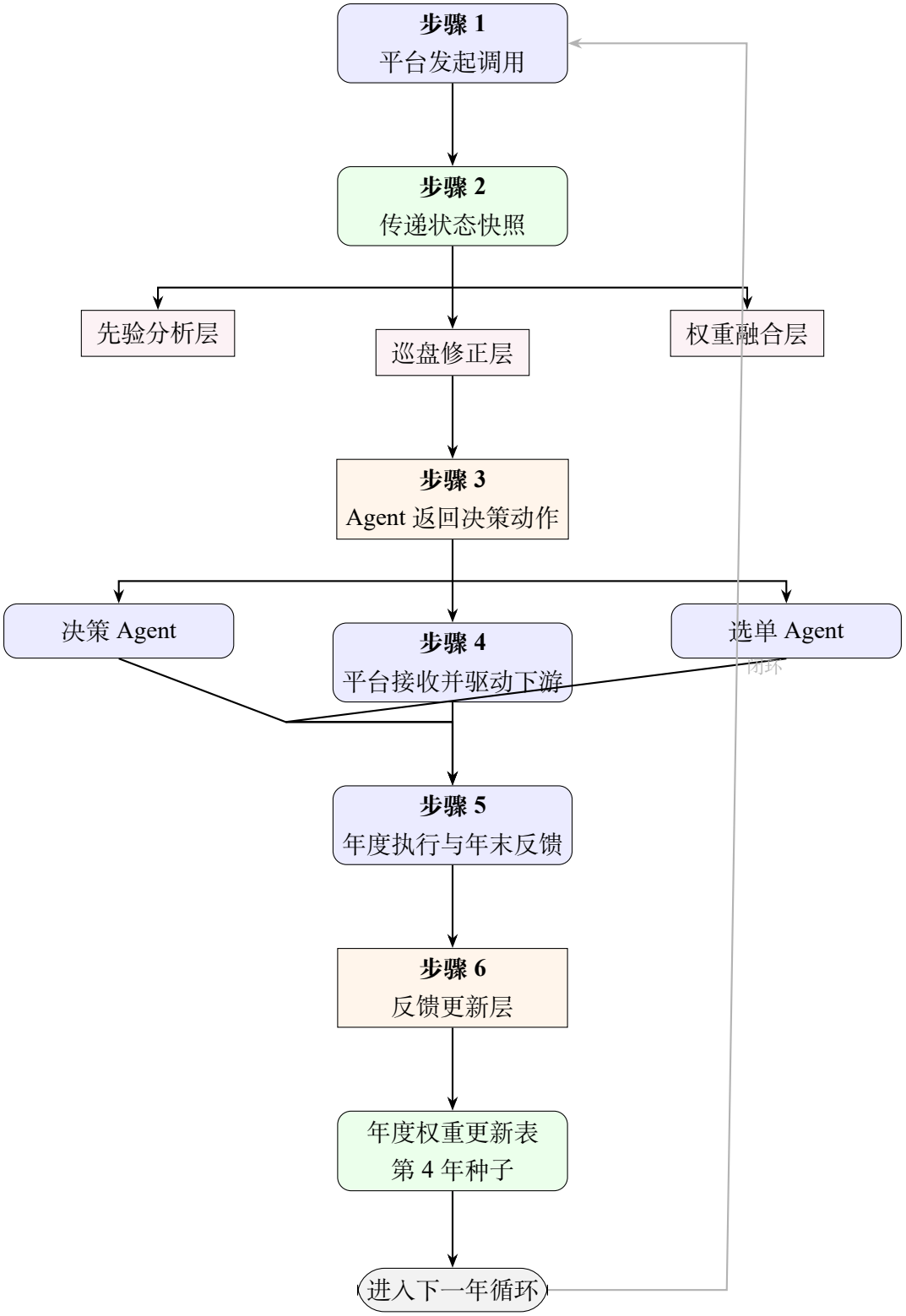

图 7: 商分 Agent 接口运行与数据接入流程示意

3.4.3 通信协议层

平台支持两种通信模式：

（1）本地进程内调用（In-Process）：Agent 与平台核心引擎运行在同一 Python 进程

中，通过函数调用直接交互。该模式延迟最低（微秒级），适用于单机构建与快速迭代，是 CLI 模式下的默认配置。

（2）远程服务化调用（Service-Oriented）：Agent 以独立 Python 进程形式部署，通过 HTTP 接口与平台核心 CLI 通信。Web 前端通过同一 HTTP API 获取比赛状态与决策结果，实现计算层与表现层的分离。该模式适用于需要将 Agent 计算与 Web 可视化分离部署的场景——例如，课题组本地计算服务器运行 Python CLI 核心引擎，阿里云服务器通过 frp 暴露 HTTP 端口供外部浏览器访问。远程调用模式下，状态快照以 JSON 序列化后通过 POST 请求传输，决策动作以 JSON 响应返回。超时机制与重试策略由平台统一管控。

3.4.4 数据格式层

状态快照（State Snapshot）与决策动作（Decision Action）均采用 JSON Schema 约束，确保各 Agent 之间的数据互操作性。关键字段定义如下：

状态快照核心字段：

1. `meta`：元信息，包含当前时间戳（year, quarter）、企业 ID、比赛轮次 ID；
2. `balance_sheet`：资产负债表，包含现金、应收款、在制品、产成品、原料、固定资产、负债等；
3. `income_statement`：利润表，包含销售收入、直接成本、折旧、维护费、管理费、财务费用、税前利润、所得税等；
4. `cashflow_statement`：现金流量表，包含经营活动、投资活动、筹资活动的现金流入流出；
5. `production`：生产状态，包含各生产线的产品、状态、剩余安装/转产时间、累计产量；
6. `market_status`：市场状态，包含各市场开拓进度、ISO 认证进度、广告投放额、订单持有情况；
7. `competitors`：竞争对手巡盘数据，包含 ProductLine 布局与 AD 矩阵（仅年初可用）。

决策动作核心字段：

1. `action_type`：动作类型枚举，如 `LONG_LOAN`、`SHORT_LOAN`、`AD_PLACEMENT`、`ORDER_SELECT`、`PRODUCE`、`RAW_MATERIAL_BUY`、`SELL_PRODUCT` 等；
2. `target`：动作目标对象，如市场 ID、产品 ID、生产线 ID；
3. `amount`：动作数量或金额；
4. `priority`：动作优先级（用于同一时点多个动作的排序）。

3.4.5 环境约束与版本锁定

为保证平台在长期运行与服务器迁移过程中的可复现性，课题组对 Python 计算环境与 Web 前端环境分别进行严格的版本锁定。

Python 计算环境（CLI 核心）。平台 CLI 核心引擎全部采用 Python 编写，便于商科学院师生在较低代码能力门槛下进行维护与迭代。采用 Conda 进行环境隔离与复现，核心依赖包括：

1. Python 3.10（LTS 版本，避免频繁升级带来的兼容性风险）；
2. pandas 2.1.x（状态表数据处理与分组聚合）；
3. numpy 1.24+（数值计算与矩阵运算）；
4. openpyxl 3.1+（Excel 规则表与状态表的读写）；
5. scipy 1.11+（优化求解与统计检验）；
6. Flask 2.3+ 或 FastAPI 0.100+（远程 Service-Oriented 模式下的 HTTP 服务）。

全部依赖版本通过 `environment.yml` 与 `requirements.txt` 双重锁定。服务器迁移时，仅需在新机器上执行 `conda env create -f environment.yml` 即可在 5 分钟内重建完整计算环境，无需手动逐包安装。

Web 前端环境。Web GUI 采用 Node.js LTS 版本（20.x，满足 Codex CLI、Kimi Code CLI 等 AI 编程工具的最小运行依赖，避免 Node 版本过旧导致安装失败），核心依赖通过 `package.json` 与 `package-lock.json` 严格锁定：

1. React 18.x + TypeScript 5.x（前端组件框架与类型安全）；
2. ECharts 5.x（数据可视化：折线图、雷达图、热力图）；
3. Axios 1.4+（HTTP 客户端，对接后端 RESTful API）；
4. Socket.io-client 4.6+（WebSocket 实时通信，支撑对战大屏低延迟刷新）。

前端构建产物（静态 HTML/CSS/JS）通过 `npm run build` 生成后部署于阿里云服务器，与后端 Python CLI 完全解耦，支持独立升级与 CDN 加速。若未来需要更换前端技术栈（比如从 React 迁移至 Vue），只需重新构建静态产物并替换 Nginx 目录，不影响后端计算核心。

3.5 预期设计效果

通过上述规则引擎化、控制流标准化、交互界面双轨化与接入协议规范化的设计，课题组预期实现以下五层效果：

（1）全自动对抗运行。在无人干预模式下，平台可连续运行完整 6 年比赛，各 Agent 按既定控制流自主完成贷款、广告、选单、生产、交货、研发、市场开拓等全部决策动作。运行结果（最终权益、总成绩、排名）稳定可复现，同一组 Agent 与同一组初始种子多次运行的结果偏差控制在 1% 以内。

（2）人机混合对抗。平台支持 Ägent vs 人类混¨合对战模式。人类选手通过 Web GUI 或传统百树客户端参与比赛，Agent 通过 CLI 或远程 API 接入。该模式是验证 Agent 策略有效性的核心场景——若 Agent 能够在与人类选手的对抗中稳定进入前三，则说明其策略具备实际竞争力。

（3）策略可视化与可解释性。通过 Web GUI 的 Agent 决策透视功能，任何一次权重输出、任何一笔广告投放、任何一个选单决策，均可追溯至其上游输入数据与下游预期

收益。这使得 Agent 不再是黑箱，而成为可审计、可教学、可复盘的策略工具。

（4）数据资产沉淀。每场比赛产生的 GST、ADC 与 EL 被自动归档至数据库，形成沙¨盘决策大数据。¨这些数据不仅可用于 Agent 的持续训练（强化学习、模仿学习），也可用于沙盘教学中的案例库建设——将优秀 Agent 的决策过程以案¨例 + 数据形¨式向学生展示。

(5) 可迁移的平台底座。平台核心引擎与百树具体规则解耦：规则表独立配置、Agent 接口抽象统一、状态机与决策流通用。这意味着，若未来需要对接其他 ERP 沙盘（如用友、金蝶）或自定义规则变体，只需替换规则表与少量适配层代码，而无需重构整个平台。