# SCML 2025 高级代理架构研究报告：HRL-X 的取证重构、主动协商机制与全球协调层深度解析

## 1. 执行摘要：从启发式规则到认知型架构的范式跃迁

供应链管理联赛（Supply Chain Management League, SCML）作为国际自动协商代理竞赛（ANAC）中最具挑战性的赛道之一，长期以来被视为检验多智能体系统（Multi-Agent Systems, MAS）在复杂经济环境中适应能力的试金石。随着2025年标准联赛（Standard League）规则的重大迭代，特别是由于非易腐库存（Non-Perishable Inventory）机制的引入、期货合约（Future Contracts）交易的开放以及更为严苛的短缺惩罚（Shortfall Penalties）体系的确立 1，竞赛的本质发生了根本性的转变。它不再仅仅是一个考察单次博弈谈判技巧的竞技场，而是演变成了一个要求代理具备长周期库存规划、跨期风险对冲以及多线程并发资源协调能力的综合性企业资源规划（ERP）模拟环境。

回顾2024年及之前的赛果，以 **PenguinAgent** 和 AS0 为代表的基于规则的启发式代理（Heuristic Agents）凭借其稳健的成本控制逻辑和确定性的风险规避策略，占据了排行榜的前列 1。然而，这种“硬编码”的智慧在面对2025年日益复杂的动态市场时显露出了明显的局限性：它们缺乏对非平稳市场环境的自适应能力，难以利用长尾分布中的投机机会，且无法有效处理由期货交易带来的状态空间指数级膨胀问题 3。为了突破这一策略天花板，学术界与工业界开始将目光投向更具认知深度的强化学习（Reinforcement Learning, RL）架构。

本研究报告深入探讨了针对SCML 2025环境定制的“混合残差学习架构”（Hybrid Residual Learner, HRL-X）。该架构并非单一算法的简单堆砌，而是一个有机的控制论系统，它通过 **分层强化学习（HRL）** 解决时间尺度的抽象与长期规划问题，利用 **决策Transformer（Decision Transformer, DT）** 处理博弈过程中的序列依赖与对手建模，并借助 **离线强化学习（Offline RL）** 中的 "Reward-on-the-Line" (ROL) 机制解决高风险环境下的探索安全与样本效率问题 1。

本报告将重点解决 HRL-X 架构实施过程中最关键、且在初步设计文档中往往语焉不详的三个核心技术难题：

1. **PenguinAgent 宏观目标的取证重构**：既然 PenguinAgent 是基于规则的黑盒，我们如何通过算法对其进行“逆向工程”，从历史交互日志中精准还原其每日的宏观库存目标（Daily Macro Goal），从而为 HRL-X 的 L2 管理层提供高质量的“专家演示”数据？
2. **协商发起的主动性悖论**：在 HRL-X 架构中，L2 负责定目标，L3 负责谈细节，L4 负责调权重。那么，究竟是由谁、通过何种机制在每个仿真日的清晨“扣动扳机”，主动向市场中的潜在对手发起协商请求（Call for Negotiation）？
3. **L4 全局协调层的本质**：这一层究竟是一个基于简单启发式规则的过滤器，还是一个具备可训练参数的神经网络？其实陈述的技术细节如何支撑其解决“并发资源耦合”这一核心痛点？

本报告将通过严谨的数学建模、代码级的逻辑推演以及深度的架构剖析，为上述问题提供详尽的解答，旨在为开发下一代超人级供应链代理提供详实的理论依据与实践蓝图。

## 2. 宏观目标的取证重构：基于 PenguinAgent 的反向演绎法

在 HRL-X 架构中，L2 战略管理者（Strategic Manager）的核心职责是在每一天开始时，基于当前的宏观状态（库存、资金、市场行情），输出当天的 **战略目标向量 (Goal Vector) $g\_t$** 1。为了训练这个 L2 模型，最理想的方法是利用“模仿学习”（Imitation Learning）或“离线强化学习”（Offline RL）来预训练网络，这就需要大量的 $(State, Goal)$ 对作为训练数据。

虽然我们拥有历年的比赛日志，但日志中只记录了代理的 **动作（Action）** ——即具体的报价（Price, Quantity），而没有直接记录代理内心的 **目标（Goal）**。对于一般的 RL 代理，其目标是隐式的、随机的，难以捕捉。但对于 **PenguinAgent** 这样基于确定性规则的代理，其行为与目标之间存在着严格的逻辑映射关系。我们可以利用这一特性，通过 **取证逆向演绎（Forensic Inverse-Deduction）**，从其行为和状态中精确重构出其每日的宏观目标。

### 2.1 PenguinAgent 的核心逻辑解构：严格量化匹配

要重构目标，首先必须理解目标的生成逻辑。PenguinAgent 的核心策略被描述为“基于精确计算的成本极小化”和“需求逆推” 1。其核心公理是 **严格量化匹配（Strict Quantity Matching）** 和 **零库存等待（Zero-Inventory Waiting）** 4。

这意味着 PenguinAgent 绝不进行投机性囤货。它在第 $t$ 天的买入目标 $Q\_{buy}^{target}$，严格等于其为了满足未来（$t+1$ 及以后）的生产承诺而在今天必须持有的原材料数量，减去当前已有的库存。

### 2.2 目标重构的数学模型

我们可以将 PenguinAgent 的每日目标向量定义为 $g\_t = [Q\_{target\\_buy}, P\_{limit\\_buy}, Q\_{target\\_sell}, P\_{limit\\_sell}]$。以下是各分量的重构算法。

#### 2.2.1 买入量目标 (**$Q\_{target\\_buy}$**) 的重构

PenguinAgent 的买入决策受到两个硬性边界的约束：**仓库容量上限**（防止爆仓成本）和 **生产需求下限**（防止违约惩罚）。

约束一：最大安全买入量（Capacity Constraint）

这是基于物理约束的硬性上限。代理绝不会计划买入超过其剩余存储能力的货物。

$$Q\_{max\\_safe} = C\_{total} - I\_{current} - I\_{incoming} + O\_{committed}$$

其中：

* $C\_{total}$：仓库总容量。
* $I\_{current}$：当前在库的原材料数量。
* $I\_{incoming}$：已签订合同但尚未交付的在途原材料（预计在生产前到达）。
* $O\_{committed}$：已签订合同需交付的成品（对应的原材料库存将被释放）。

约束二：最小必要买入量（Demand Constraint）

这是基于“需求逆推”逻辑计算出的刚性需求。PenguinAgent 追求 JIT（准时制）生产，因此它只买未来订单急需的量。

假设代理在未来 $H$ 天（PenguinAgent 的视野通常很短，专注于近期交付）有 $K$ 个销售合同，每个合同需求量为 $D\_k$。

$$Q\_{needed} = \max\left(0, \sum\_{k \in \text{Window}(H)} D\_k - I\_{current} - I\_{incoming}\right)$$

重构公式：

由于 PenguinAgent 是极度风险厌恶的，且追求零库存，其实际设定的宏观目标 $Q\_{target\\_buy}$ 通常就是这两个约束的交集。在大多数非极端情况下，为了满足订单并避免违约，它会倾向于买入 $Q\_{needed}$，但绝对受限于 $Q\_{max\\_safe}$。

$$Q\_{target\\_buy}^{reconstructed} = \min(Q\_{max\\_safe}, Q\_{needed})$$

如果日志显示当天没有未来的销售合同（或者处于游戏初期），根据 PenguinAgent 的保守策略，其 $Q\_{target\\_buy}$ 可能为 0，或者是一个极小的试探性数值（用于维持市场活跃度）。但根据文献 1，它“绝不进行投机性囤货”，因此在无订单时，$Q\_{target\\_buy} \approx 0$ 是一个高置信度的重构假设。

#### 2.2.2 买入限价目标 (**$P\_{limit\\_buy}$**) 的重构

PenguinAgent 的定价策略是为了保本微利。它不会为了买入而支付导致最终亏损的价格。

**重构逻辑：**

1. **获取生产成本**：$C\_{process}$（加工费，由工厂私有配置决定，但在训练数据中可知）。
2. **获取市场参考价**：$P\_{market\\_sell}$（成品在公告板上的平均成交价，或历史销售均价）。
3. 逆推盈亏平衡点：  
     
   $$P\_{break\\_even} = P\_{market\\_sell} - C\_{process}$$
4. 留出安全边际：PenguinAgent 通常会预留一个最小利润率 $\lambda$（例如 5% - 10%）。  
     
   $$P\_{limit\\_buy}^{reconstructed} = \frac{P\_{market\\_sell}}{1 + \lambda} - C\_{process}$$  
     
   或者更简单地，基于历史观测，它可能直接在成本线上加成：  
     
   $$P\_{limit\\_buy}^{reconstructed} \approx P\_{history\\_avg} \times (1 - \text{Discount\\_Factor})$$

通过分析 negotiations.csv 日志中 PenguinAgent 在同一天发出的所有 Offer，找到其 **最高出价（Max Offer Price）**，该值通常非常接近其内部设定的 $P\_{limit\\_buy}$。我们可以取当天所有谈判线程中的最高出价作为 $P\_{limit\\_buy}$ 的标签。

#### 2.2.3 卖出目标的重构

同理，卖出目标受限于当前的可售库存。

$$Q\_{target\_sell}^{reconstructed} = I\_{finished\_goods} + I\_{production\_today}$$卖出底价 $P\_{limit\\_sell}$ 则通常基于成本加成：

$$P\_{limit\\_sell}^{reconstructed} = (P\_{raw\\_avg} + C\_{process}) \times (1 + \lambda)$$

### 2.3 取证重构的实施流程

为了为 HRL-X 的 L2 层构建数据集，我们需要执行以下标准化流程 4：

1. **数据清洗（Data Ingestion）**：
   * 读取 SCML 历年比赛的 negotiations.csv 和 world\_stats.csv。
   * 筛选出所有 agent\_id 为 PenguinAgent 的记录。
2. **状态重建（State Reconstruction）**：
   * 对于每一个仿真日 $t$，从日志中提取当天的快照状态 $S\_t$：库存水平、资金余额、已签订的合同队列、市场指数。
3. **目标标签化（Goal Labeling）**：
   * 利用上述 2.2 节的公式，结合当天的状态 $S\_t$，计算出理论上的 $g\_t$。
   * **校准（Calibration）**：检查当天 PenguinAgent 实际达成的交易总量 $Q\_{executed}$。如果 $Q\_{executed} \approx Q\_{target}^{reconstructed}$，则标签可信。如果偏差较大，说明可能触发了某种紧急规则（如临近破产保护），需引入修正项。
4. **数据集构建（Dataset Compilation）**：
   * 生成样本对 $(S\_t, g\_t)$。
   * 这些样本将被用于 **监督学习（Supervised Learning）**，训练 L2 决策 Transformer 的初始权重，使其学会像 PenguinAgent 一样制定稳健的库存规划。这将解决 RL 训练初期的“冷启动”问题，防止代理因瞎指挥而迅速破产。

## 3. 协商发起机制：L2 与 L3 之间的“缺失环”

用户提出的第二个问题切中了 HRL-X 架构描述中的一个关键空白：**主动性（Proactivity）**。

* L2 只负责说“今天要买 50 个”。
* L3 只负责在谈判桌上说“我出 95 元”。
* L4 只负责说“3号谈判线程更重要”。

那么，是谁去敲门？是谁在早晨拨通供应商的电话？如果代理只是被动等待（Respond），在竞争激烈的 SCML 市场中将面临极其严重的“长尾饥饿”——即根本收不到足够的 Offer 来满足 L2 设定的 50 个购买目标。

因此，HRL-X 架构中必须包含一个隐式的、但至关重要的 **“发起层”（Initiation Layer）**。基于 SCML 的 API 规范 5 和 HRL-X 的设计哲学，我们提出并详述 **“广播-过滤”（Broadcast-Filter）** 协议作为其标准实现。

### 3.1 为什么必须是“广播”策略？

在 L2 设定了 $Q\_{target} = 50$ 之后，代理面临一个经典的 **探索-利用（Exploration-Exploitation）** 困境：

* 应该只找一个老朋友谈 50 个？（风险：对方可能缺货或要价过高，导致全天任务失败）
* 应该找 5 个供应商，每人谈 10 个？（风险：碎片化成交，且可能部分失败）
* 应该找所有供应商，每人谈 50 个？（风险：**过度承诺（Over-commitment）**，如果大家都同意，就会买入 500 个，导致瞬间爆仓破产）。

对于 PenguinAgent 这样的规则代理，它通常采用保守的轮询策略。但对于 HRL-X 这样的智能代理，为了最大化捕捉市场中的低价机会，最佳策略是 **激进的广播（Aggressive Broadcast）**。

### 3.2 具体的发起流程（Day Start Workflow）

这一过程发生在 SCML 仿真步的 before\_step() 阶段，位于 L2 决策之后，L3 执行之前。

#### 步骤一：L2 目标解析

代理接收 L2 输出的目标向量 $g\_t$。

假设 $Q\_{target\\_buy} = 50$, $P\_{limit\\_buy} = 100$。

#### 步骤二：目录检索与伙伴筛选

代理调用 awi.my\_suppliers 获取当前市场中所有能够提供所需原材料的供应商列表。

$$List\_{partners} = \{Ag\_1, Ag\_2, \dots, Ag\_N\}$$

此时，代理可以应用一个简单的 基于声誉的过滤器（Reputation Filter）（这是 L2 的一部分或独立的小模块），剔除那些历史违约率极高的黑名单代理。

#### 步骤三：饱和式协商请求（The Saturation Request）

这是关键动作。代理向 $List\_{partners}$ 中的 每一个 供应商都发起一个新的协商请求。

调用 API：awi.request\_negotiation(partner=Ag\_i, product=raw, quantity=Q\_target\_buy,...) 6。

**注意**：这里请求的数量是 $Q\_{target\\_buy}$（50个），而不是 $Q/N$。这意味着代理向市场发出了总计 $N \times 50$ 的购买意向。

* **目的**：最大化接触面，确保不错过任何一个可能的低价卖家。
* **风险**：严重的过度承诺风险。

这就引出了 HRL-X 架构中 L4 存在的根本原因——如果没有 L4，这种广播策略就是自杀；有了 L4，这就是最高效的市场探测手段。

#### 步骤四：L3 线程实例化

当这些请求被对方接受（或对方发起的请求被我方接受）时，SCML 模拟器会为每一对协商生成一个唯一的 negotiation\_id。

HRL-X 为每一个活跃的 negotiation\_id 实例化一个 L3 Residual Actor（决策 Transformer）。

此时，我们可能有 20 个并行的 L3 线程，每个都在试图买入 50 个单位。

### 3.3 为什么 L2/L3 不适合做选择？

* **L2 不适合**：L2 运行在“天”的尺度上。如果让 L2 决定“今天只跟 Agent A 谈”，一旦 Agent A 发生意外（如破产、缺货），L2 要等到第二天才能修正，这一天的产能就浪费了。L2 的决策粒度太粗。
* **L3 不适合**：L3 是局部的。每个 L3 实例只看得到自己的对手。L3（线程 A）不知道 L3（线程 B）已经谈到了一个极低的价格。让 L3 做选择会导致局部最优。

因此，**“广播-过滤”** 协议将“选择权”实际上移交给了 **L4 全局协调层**。发起层负责“广撒网”，L4 负责“重点捕鱼”。

## 4. L4 全局协调层：从启发式到注意力机制的进化

用户核心疑问：L4 到底是什么？是简单的 if-else 规则，还是需要训练的神经网络？

答案是肯定的：L4 是一个基于注意力机制（Attention Mechanism）的可训练神经网络层。它绝非简单的启发式规则 3。

### 4.1 为什么启发式 L4 会失效？

如果是基于规则的启发式 L4，逻辑可能是：“优先选择价格最低的线程”。

失效场景：

* **陷阱报价**：对手 A 报了一个极低价，但这是个“钓鱼”行为，它会在最后时刻撤回或故意违约。启发式规则会早早地把权重全给 A，导致忽略了诚实的对手 B。
* **战略压制**：有时为了维持与关键大客户的关系（长期利益），或者为了挤压竞争对手，我们可能需要优先与价格稍高的对手成交。简单的规则无法捕捉这种高阶博弈逻辑。

### 4.2 L4 的技术实现：集中式注意力网络

L4 的核心是一个 **多头自注意力（Multi-Head Self-Attention, MHSA）** 模块，类似于 Transformer 中的 Encoder Block。

#### 4.2.1 输入张量架构

L4 接收来自所有活跃 L3 线程的隐状态（Hidden States）。

假设当前有 $K$ 个活跃谈判。每个 L3 Decision Transformer 输出一个隐向量 $h\_k \in \mathbb{R}^{d\_{model}}$。

L4 的输入是一个集合张量 $H\_{in} = \{h\_1, h\_2, \dots, h\_K\}$。

此外，L4 还需要接收一个 全局上下文向量（Global Context Vector） $C\_{global}$，包含当前的 L2 目标 $g\_t$、剩余资金、剩余总需求量。

#### 4.2.2 核心计算过程

L4 通过 Attention 机制计算每个线程对实现当前全局目标的“重要性”。

1. **Query-Key-Value 投影**：
   * **Query ($Q$)**：来自全局上下文 $C\_{global}$。代表“当前系统最急需解决的问题是什么？”（例如：急需补货，不管价格）。
   * **Keys ($K$)**：来自各个 L3 的隐状态 $h\_k$。代表“该谈判线程当前的状况特征”（例如：对手让步很快，价格适中）。
   * **Values ($V$)**：同样来自 $h\_k$。
2. 注意力权重计算：  
     
   $$Attention(Q, K) = \text{Softmax}\left(\frac{Q K^T}{\sqrt{d\_k}}\right)$$  
     
   得出的权重向量 $\alpha = [\alpha\_1, \alpha\_2, \dots, \alpha\_K]$ 就是每个谈判线程的 优先级权重。
   * 若 $\alpha\_i$ 很高，说明线程 $i$ 是当前最有可能、最高效帮助实现 L2 目标的途径。
3. 门控输出（Gating Output）：  
   L4 不仅仅输出权重，它通过一个门控机制直接调节 L3 的策略分布。  
     
   $$Action\_{L3}^{final} = \text{Policy}\_{L3}(h\_k | \alpha\_k)$$
   * **高权重 ($\alpha\_k \to 1$)**：L3 变得 **激进（Aggressive）**，倾向于接受当前 Offer 或仅做微小让步以确保成交。
   * **低权重 ($\alpha\_k \to 0$)**：L3 变得 **保守（Conservative）** 或 **拖延（Stalling）**，报出极具侮辱性的低价或直接拒绝。这实际上就是 L4 在执行“过滤”——虽然连接建立了，但我不打算真的跟你成交，除非你给出一个无法拒绝的价格。

### 4.3 L4 的训练机制：端到端的多智能体强化学习

L4 这一层 **必须学习**，且必须与 L3 协同训练。它不是预先定义好的。

训练算法：MAPPO (Multi-Agent PPO) 3

HRL-X 的训练采用 集中式训练，去中心化执行（CTDE） 的范式。

* **Critic（评论家）**：在训练阶段，Critic 能看到全局信息（所有 L3 的状态 + L4 的权重 + 最终的全局奖励 $R\_{total}$）。
* **全局奖励信号**：$R\_{total}$ 反映了当天的总利润和库存健康度。
* 梯度反向传播：  
  如果 L4 给一个“骗子”对手分配了高权重，导致最终违约或亏损，全局奖励 $R\_{total}$ 会变低。  
  通过 PPO 算法，梯度会反向传播回 L4 的 Attention 参数矩阵 ($W\_Q, W\_K$)。  
  结果：L4 学会了识别那些“表面诱人但实际有毒”的谈判模式，并自动降低其权重。

**结论**：L4 是 HRL-X 架构中的“指挥家”。它通过学习将有限的资源（资金、购买配额）动态分配给最有价值的谈判线程，解决了“广播”策略带来的过度承诺风险。它是一个深度神经网络层，其智能来自于海量博弈数据的训练，而非人工规则。

## 5. 总结：HRL-X 的完整运行图景

结合上述三个部分的深度剖析，我们可以勾勒出 HRL-X 代理在 SCML 2025 中的完整且具体的运行图景：

1. **黎明时刻 (Daily Start)**：
   * **数据流**：历史日志被用于重构 PenguinAgent 的逻辑，L2 Strategic Manager 经过预训练后，读取当前的高维市场状态。
   * **决策**：L2 输出精确的宏观指令：“今日需买入 50 单位，均价不高于 105”。
2. **早晨 (Initiation Phase)**：
   * **动作**：执行 **“广播-过滤”协议**。代理不进行主观挑选，而是向所有 20 个潜在供应商广播购买 50 单位的意向（Total Request = 1000）。
   * **目的**：确保最大的市场接触面，捕捉任何可能的长尾低价。
3. **日间 (Negotiation Loop)**：
   * **并发处理**：20 个 L3 Residual Actor 线程同时启动。
   * **实时协调**：**L4 全局协调层**（Attention Network）实时监控这 20 个线程的隐状态。
   * **动态过滤**：L4 发现其中 3 个供应商反应积极且价格合理，赋予其高权重 $\alpha$；其余 17 个线程被赋予低权重。
   * **战术执行**：
     + 高权重 L3：迅速接受报价或微调成交，锁定了 50 个单位。
     + 低权重 L3：报出极低价。如果对方接受，则那是意外之喜（套利）；如果对方拒绝，则无伤大雅，因为主要目标已由高权重线程完成。
4. **日暮 (Settlement)**：
   * **结果**：代理成功买入约 50 个单位（误差极小），均价优于市场平均。
   * **反馈**：MAPPO 算法根据最终利润更新 L2、L3 和 L4 的参数，使明天的协调更加精准。

通过这种 **“取证重构定基调、广播机制保触达、注意力网络做筛选”** 的严密逻辑闭环，HRL-X 架构成功地将规则的确定性与 AI 的适应性融为一体，为在 SCML 2025 标准联赛中实现超人表现奠定了坚实的技术基础。

## 6. 数据表格与参数附录

### 表 1: PenguinAgent 目标重构公式汇总

| **目标分量** | **重构公式** | **逻辑依据** |
| --- | --- | --- |
| **买入量** $Q\_{target\\_buy}$ | $\min(C\_{total} - I\_{curr} - I\_{in} + O\_{out}, \sum D\_{future} - I\_{net})$ | 物理容量约束 $\cap$ 需求逆推 (JIT) |
| **买入限价** $P\_{limit\\_buy}$ | $\min(P\_{offer}^{max\\_daily}, P\_{mkt\\_sell} - C\_{prod} - \text{Margin})$ | 历史最高出价逼近 $\cup$ 盈亏平衡点逆推 |
| **卖出量** $Q\_{target\\_sell}$ | $I\_{finished} + I\_{production\\_today}$ | 清空库存逻辑 (Zero Inventory) |
| **卖出底价** $P\_{limit\\_sell}$ | $(P\_{raw\\_avg} + C\_{prod}) \times (1 + \lambda)$ | 成本加成定价法 (Cost-Plus) |

### 表 2: L4 全局协调层技术规格

| **特性** | **描述** |
| --- | --- |
| **核心算法** | 多头自注意力机制 (Multi-Head Self-Attention) |
| **输入维度** | $(Batch, K\_{threads}, d\_{model})$ |
| **可训练参数** | $W\_Q, W\_K, W\_V$ (投影矩阵), Feed Forward Networks |
| **输出信号** | 线程重要性权重 $\alpha \in ^K$ |
| **训练方法** | MAPPO (End-to-End with L3) |
| **作用机制** | 调节 L3 策略熵（Aggressiveness/Conservatism） |

#### 引用的著作

1. HRL-X 研究：强化学习问题解决
2. SCML 2024 Standard Track Winner: PenguinAgent - YouTube, 访问时间为 十二月 9, 2025， <https://www.youtube.com/watch?v=ueXgfjpXuFI>
3. SCML 代理开发技术架构
4. HRL-X 代理实现与数据收集
5. Developing an agent for SCML2024 (Standard) - scml 0.7.7 documentation, 访问时间为 十二月 9, 2025， <https://scml.readthedocs.io/en/master/tutorials/04.develop_agent_scml2024_std.html>
6. Overview — NegMAS 0.10.18 documentation, 访问时间为 十二月 9, 2025， <https://negmas.readthedocs.io/en/v0.10.18/overview.html>