

下面是一份系统级、端到端的对照与体检报告。我把 Presentation/Script 里的系统设计（四 Agent + 路由 + 可观察 + 配置驱动）与当前实现逐条对齐，指出已完成、差距、风险与下一步验证/实验建议。你可以把它当成一次“上线前的技术评审”。

1. 总览结论（TL;DR）

- ****架构一致性****：已实现“Flow 适配层 + Impl 实现层”的插件化模式；四大 Agent（GraphConstruction / Retrieval / Reasoning / Verification）与路由器（LLMRouter）、工厂（factory）、统一配置（settings.yaml）和可观测（TelemetrySink）对齐。
- **关键算法路径**：
 - GraphConstruction：支持自举上下文（Bootstrap via Retrieval）、NodeBuilder / EdgeBuilder、NetworkX 落盘；
 - Retrieval：支持 Hybrid（LLM Query Expand + BM25 + Graph 邻域 + 融合），也可走内置 BM25 Flow；
 - Reasoning：Planner + Synthesizer（计划 → 证据引用 → 综合）；
 - Verification：规则（覆盖率/引用）+ LLM 核验 双通道，统一输出。
- **主要缺口（建议优先级 P0/P1）**：
 1. **检索评估与可重复性（P0）**：缺标准指标与实验脚本（Recall@k、MRR、Graph 召回）；
 2. **Reasoning 受控引用（P0）**：需确保只用已检索证据生成，强约束/模板化提示；
 3. **Rerank / Dense 检索（P1）**：当前缺强力重排与向量召回；
 4. **Prompt/Policy 版本化（P1）**：Prompt 文档化、哈希与 A/B 管理；
 5. **实验基线与报告（P1）**：HotpotQA 的 EM/F1、Evidence F1、成本/延迟报表；
 6. **缓存与资源控制（P1）**：LLM 调用、索引与图加载的缓存；
 7. **YAML Schema 校验（P2）**：防止 impl_kwargs 混入 flow 参数再复发。

下面进入细节核对与建议（含“已达成/待完善/建议实现”三态）。

2. 架构与依赖注入

目标（PPT）：模块化、可替换、配置驱动；Flow 只协调，Impl 执行业务；router/sink 可注入；settings 统一。

当前实现：

- `factory.py` 统一解析 `modules.*`，将 `impl/impl_kwargs` 注入 **Flow**，并以 `(settings, router, sink)` 调用 `from_settings()`；
- Flow 层（四个模块）已统一签名并负责实例化 `Impl`（含反射过滤 + 自动注入 `router/sink`）；
- 解决了历史问题：`sink` 形参不匹配、`impl_kwargs` 混入 `flow` 参数、`HybridBackend` 缺少 `retrieve()` 等。

✅ 一致：建议保留一份《装配约定》简表（谁接收什么参数），并在 CI 做配置校验（见“工程实践”）。

3. GraphConstruction（建图）

1. 设计目标（PPT）

- 输入仅有 **question** 也能自举获取 **context**，生成 **Nodes + Edges**；
- `NodeBuilder/EdgeBuilder` 策略可配；
- `Impl`（如 `NetworkX`）负责落盘（`json/gexf/manifest`）。

2. 代码现状

- **Flow**：编排 `Ingest` → `BootstrapContext` → `BuildNodes` → `BuildEdges` → `AssembleSave`；
- **BootstrapContext**：当无 `context` 时，调用 **RetrievalAgentFlow** 依问题检索，按 `doc -> [sent]` 聚合给 `NodeBuilder`；
- **NodeBuilder / EdgeBuilder**：按策略生成句子/文档/实体节点与多类边；
- **Impl（NetworkX）**：保存至 `data/graph/<graph_id>/{json,gexf,manifest}`；
- **配置**：`modules.graph_construction.impl / impl_kwargs`（只传给 `Impl`），`node_builder / edge_builder`（传 `Flow`）。

3. 评估

- ✅ 达标：支持“只给 `question` 也能建图”；
- ⚠️ 需要明确/验证的点：
 - 语义边（`use_semantic_edges`）若存在，需要 **embedding provider**（已在 `llm_policy.embedding_provider` 指定）；检查 `edge_builder` 是否已接入并在大文件上跑过；
 - 图规模控制：`Bootstrap` 的 `top_k`、图邻域窗口 `graph_window` 是否可随任务动态调整？
 - 图一致性：节点/边 ID 命名规范及去重策略（防止重复写入/多源合并时冲突）。

4. 建议

- **指标**：记录每次建图的 `node_count/edge_count/类别占比/保存时延`；
 - **回归用例**：给三类输入场景固定检查（仅 `question` / 有 `context` / 带自定义 `nodes/edges` 合并）；
 - **冷/热启动**：对常见 `graph_id` 做缓存与一致性校验（`manifest` 校验和）。
-

4. RetrievalAgent（检索）



5. 设计目标（PPT）

- LLM Query Expand + 文本检索 + 图邻域 + **融合排序**；可替换后端；配置驱动。

6. 代码现状

- **Flow**：支持 `backend` 外部实现（`HybridRetrievalBackend`），否则走内置 `BM25 + Graph + 融合 LangGraph`；
- **HybridRetrievalBackend**：`expand → text (BM25Lite) → graph → fuse`，并新增标准 `retrieve()` 输出 `RetrievalOut`；
- **配置**：`modules.retrieval.kwargs (id/score key、窗口/权重...)`，`impl/impl_kwargs (index_path, graph_root, top_k...)`。

7. 评估

-  **达标**：hybrid 管线与接口统一；
-  **缺口/待验证**：
 - **Reranker**（Cross-Encoder）缺失；
 - **Dense 向量检索**未接入（仅 `BM25 + Graph`）；
 - 归一化与去重策略需明确（跨源合并分值、同文本不同 `id`）；
 - **可重复性**：LLM 扩展的随机性（需温度=0 + 缓存/种子/提示模板稳定化）。

8. 建议

- 引入可选 **Nomic/Contriever/ColBERT** 向量召回；
 - 二阶段 **Cross-Encoder 重排**（如 `bge-reranker / E5-rerank`）；
 - 评测脚本：**Recall@k / MRR / 命中覆盖率**（对 HotpotQA 支持证据的召回率），并与纯 `BM25` 做 A/B。
-

5. ReasoningAgent（推理）



9. 设计目标（PPT）

- 计划（**plan**）→ 综合（**synthesize**）；答案带引用 [#k]；对证据进行裁剪、对齐和引用。

10. 代码现状

- **Flow**: 适配 `impl_planner_synth`;
- **Impl (Planner+Synth)**: 规划步骤、引用块构造、答案综合;
- **输出**: `ReasoningOut(answer, evidence_used, steps, model)`（引用格式 [#k]）。

11. 评估

-  **达标**: Planner+Synth 路径清晰，输出对 Verification 友好;
-  **缺口/待验证**:
 - **受控引用**: 需要保证 Synth 只引用 `evidence_used`，避免幻觉引用（可在系统提示中强约束，并对 [#k] 进行正则验证与回填）;
 - **多样性/自洽**: 若有多草稿投票/自洽（**self-consistency**），需要记录每次候选与票数（`steps` 中已有雏形，可标准化）;
 - **检索-推理迭代**: 当前是单轮检索 → 推理，可选实现检索失败时的**回退/再检索**（基于 `plan` 子问题）。

12. 建议

- 将 **Prompt 模板** 常量化（带版本号与哈希），纳入 `llm_policy.routes`;
- 增加 `max_citations` 与 `min_citations` 控制，减少无根据长段输出;
- 引入“**证据高亮/对齐**”辅助功能（把 [#k] 映射回文本片段，便于 UI 展示与审核）。

6. VerifierAgent（核验）



13. 设计目标（PPT）

- 规则 + LLM 双通道；可配置阈值；统一输出；为最终回答与回路纠错提供依据。

14. 代码现状

- **Flow**: 适配器完成, 支持 (settings, router, sink) 与 impl 注入;
- **Impl (Rules+LLM)**:
 - 规则: 空答、是否存在 [#k]、**coverage** (被引用的证据占比)、引用数量门槛;
 - LLM: 只基于给定证据进行事实核验, **强制 JSON** 输出 (score/verdict/issues/used), 健壮解析;
 - 汇总分: $0.4 * rule + 0.6 * llm$, 输出 `VerifyOut(status, ok, score, issues, diagnostics, model)`;
- **配置对齐**: `require_citation_in_answer / min_citations / min_coverage_ratio / temperature / ctx`.

15. 评估

-  **达标**: 规则与 LLM 结合、输出结构标准化;
-  **建议**:
 - 将权重与阈值暴露到配置, 并在 Telemetry 中记录**命中原因** (哪条规则触发、LLM verdict);
 - 考虑 NLI/Entailment 模型作为 **第二条 LLM-free 通道**, 避开模型波动。

7. LLM Policy / 路由

目标 (PPT): 路由按模块/子任务映射到不同模型; 同一供应商 (OpenAI) 可统一; 成本追踪。

现状:

- 你已统一到 gpt-4o 或 gpt-4o-mini;
- price 用于成本标注/路由选择;
- embedding_provider: openai 与检索/语义边一致。

建议:

- 如果要区分输入/输出价, 可扩展 price_in/price_out 并更新 Router;
- 给每个 route 的 prompt 模板写版本号 (prompt_ver: vYYYYMMDD_x)。

8. 配置 (settings.yaml)

目标： 单一入口、无重复字段、清晰的模块边界。

现状：

- `modules.{graph_construction,retrieval,reasoning,verification}` 结构统一；
- `impl/impl_kwargs` 与 `kwargs` 已分离（Flow/Impl 各管其参）；
- 避免了 `node_builder/edge_builder` 误入 `impl_kwargs` 的问题。

建议：

- 引入 **JSONSchema/YAML schema** 校验（pre-commit 钩子），防止再次出现“unexpected keyword”类错误；
 - 对 key 集合（`id_keys/score_keys`）与检索窗口等做**默认值检查**与日志吐出。
-

9. 可观测/稳定性

目标： 端到端 trace、成本/延迟可见、错误可定位。

现状：

- `TelemetrySink` 已接入 Flow/Impl 节点跨度（span）；
 - 关键修复： `from_settings sink` 形参一致；
 - 建议在每条路由调用处记录：模型、token 数、价格字段、延迟；
 - 建议添加 **LLM/检索缓存**（基于 Prompt+输入哈希），减少费用与方差。
-

10. 实验与评测（强烈建议补齐）

为了证明“实现达成了设计目标”，需要最少的可复现实验与报表。

16. 任务与数据

- HotpotQA（已有 `ingest_hotpotqa.py`）。

17. 指标建议

1. 检索：Recall@k、MRR、命中覆盖率（支持证据被召回比例）、图邻域召回率；
2. 推理：EM / F1；答案 `[#k]` 的对齐正确率；

3. 核验：通过率 / 拒绝率；规则分 vs LLM 分的散点图；
4. 性能：端到端延迟、各阶段占比；
5. 成本：按路由与模块拆分的 token 用量与估算费用。

18. 基线 & 消融

- Baseline：纯 BM25 + 简单 prompt；
- Ablation：去掉 Graph、去掉 Query Expand、去掉 Rerank（后续加上）、不同 alpha_text/alpha_graph、不同窗口/TopK；
- 两套 policy：gpt-4o-mini (16k) 与 gpt-4o (128k)。

19. 脚本/产物

- run_system.py 增加 --eval hotpotqa --subset dev[:100] --policy mini|full；
- 输出统一到 reports/YYYYMMDD/...：JSON 指标、CSV 明细、Markdown 汇总、失配样例（方便误差分析）。

11. 工程实践与风险清单

- P0
 - 检索评测脚本与可重复实验（温度=0、缓存、固定随机源）；
 - Reasoning 受控引用（只允许 [#k] 指向 evidence_used，越界则回退/重试）；
- P1
 - Dense 检索与 Reranker；
 - Prompt/Policy 版本化、A/B 管理；
 - 统一错误处理与回退策略（LLM 超时/空输出时的 degrade plan）；
- P2
 - YAML schema + CI 校验；
 - 数据与索引缓存（BM25 索引、Graph manifest 校验、LLM 缓存）。

12. 最终结论

- 核心能力已经落地：四个 Agent、路由/工厂、配置与可观测均与设计对齐，关键路径可运行；
- 要达到“完全实现设计目标”的发表/上线标准，还需补齐：
 1. 实验体系与指标（证明有效与稳定），
 2. 受控引用与可重复性（提升可靠性与可解释），

3. 检索强化（Dense + Rerank）与 Prompt/Policy 管理（工程化）。

如果你愿意，我可以直接给出：

- 一份 **HotpotQA** 评测脚本（含指标产出与报表模板）；
 - **Prompt** 模板库（带哈希与路由映射）；
 - **YAML Schema** 与 **pre-commit** 配置（防回归）。
- 说一声，我就把对应代码/文件一次性贴出来。