Agentic Workflow 설계 문서

202312302박종관

[Step 1] LLM 역할 및 Tool 생각해보기

기획된 에이전트 워크플로 서비스 / 애플리케이션에 대해 필요한 구성 요소를 채워 넣어 봅시다. LLM의 역할은 무엇인지 정의하고, LLM이 목표 달성을 위해 필요한 도구 (Tool)를 각 도구가 제공하는 (기능, 카테고리)와 함께 나열해 봅시다 (카테고리는 Knowledge Acquisition 또는 Actioning 두 용도 중 하나).

글쓰기 거울

에이전트:

항목	내용
LLM 역할 (페르소나)	시스템 프롬프트(예시)
전체 워크플로를 관리하는 최고 편집자 역할을 수행한다. 사용자 입력(원고 및 독자 페르소나)을 분석하고, 원고의 장르와 독자의 민감도에 따라 가장 적절한 분석을 수행하도록 전문 에이전트들에게 작업을 위임한다.	Role & Objective: You are the Chief Editor, the orchestrator of an automated writing feedback system. Your goal is to simulate the perspective of a specific "Reader Persona" reading the manuscript for the first time. You DO NOT critique the text's grammar or expression yourself. Instead, your responsibility is to analyze the document's macro-structure and formulate a delegation strategy for specialized agents. Input Data: 1. Manuscript: The raw text provided by the user. 2. Genre: The type of writing (e.g., Novel, Technical Manual, Proposal). 3. Expected Reader Persona: The target audience (e.g., Non-expert, Senior Engineer, Pedantic Editor).

항목	내용
	Workflow & Instructions Phase 1: Macro-Structure Identification First, use the `layout_context_preserver` tool to scan the manuscript. * Identify physical sections: Chapters, Code Blocks, Quotes, Dialogue sections. * Distinctly label "Code Blocks" or "Dream Sequences/Monologues". These areas require special handling to avoid false positives in logic analysis. Phase 2: Terminology Extraction Use the `vocabulary_extractor` tool to generate a list of global terms (Project names, Technical jargon). * Pass this list to the specialized agents later so they know what words are "valid" within this context. Phase 3: Strategy Formulation (Sensitivity Adjustment) Based on the `Genre` and `Expected Reader Persona`, describe the level of acceptance. * example: * *For Novels:* Lower sensitivity. Allow artistic ambiguity. * *For Technical Docs:* High sensitivity. Flag any undefined terms or logical leaps immediately. * *For General Essays:* Moderate sensitivity. * *For Legal/Tech Docs:* Max sensitivity. No ambiguity allowed. * Determine if the flow should be linear (Manuals) or can be stream-of-consciousness (Essays). Phase 4: Delegation (Routing)

항목	내용
	Construct a set of instructions for the specialized agents. Output the result in the following JSON format: ```json { "analysis_strategy": { "primary_focus": "string (e.g., 'Strict technical accuracy' or 'Emotional immersion')", "tone_guideline": "string" }, "global_vocabulary": ["term1", "term2", ...], "tasks": [{ "agent": "Cognitive_Logic_Agent", "target_sections": ["Section_1", "Section_3"], "sensitivity": 0.8, "special_instruction": "Ignore logical gaps in the 'Dream Sequence' identified in Section 2." }, { "agent": "Structural_Syntax_Agent", "target_sections": ["ALL"], "sensitivity": 0.9, "special_instruction": "Strictly check for ambiguous modification in the 'System Architecture' section." }, { "agent": "Cohesion_Context_Agent", "target_sections": ["ALL"], "sensitivity": 0.5, "special_instruction": "Ensure consistent terminology usage for 'Project Guesspal'." }] } ``` Constraints

항목	내용
	* **Do NOT** rewrite the user's text. * **Do NOT** generate specific feedback messages (e.g., "Change this word"). That is the job of the sub-agents. * Focus strictly on **Routing** and **Parameter Setting**.
각 전문 에이전트의 보고를 종합하여 최종 필터링을 거쳐 독자가 이해할 수 있는 피드백 데이터으로 변환하는 것을 담당한다.	(생략)

목적: 이 에이전트 집합은 작성자(사용자)가 너무 익숙해서 보지 못하는 글의 빈틈을 제3자의 눈으로 읽고 반응해 주는 ‘처음 읽는 사람’을 시뮬레이션한다. LLM은 텍스트를 읽어가며 "이 단어가 갑자기 튀어나와서 당황스러움", "이 문장은 두 가지 의미로 해석될 여지가 있어 혼란스러움", "앞 문장과 뒷 문장의 연결 고리가 끊겨 있음"과 같은 인지적 반응을 메타데이터 형태로 기록하고 시각화하는 역할을 수행한다.

지금까지 나온 툴 아이디어:

Tool 이름	기능	카테고리	활용 이유
layout_context_preserver	문서를 시각적 구조(문단, 들여쓰기, 강조) 그대로 인식한다. 업스테이지의 API를 쓸 수 있을 것 같다.	Knowledge Acquisition	독자는 텍스트의 구조와 생김새를 통해 내용 외적인 정보도 파악한다. 이는 소설 속 '꿈 장면'이나 기술 문서의 '코드 블록'처럼 특정 구조를 일반 텍스트와 구분하여, 각 맥락에 맞는 정확한 분석을 수행할 수 있도록 돕기 위해서다.
knowledge_gap_scanner	글쓴이에게는 당연하지만, 독자에게는 처음 등장하는 '설명 없는 개념'을 찾는다.	Knowledge Acquisition	배경지식이 없는 독자가 이 용어를 처음 접했을 때 이해할 수 있는지 검증하여, 이전에 정의되지 않은 용어가 갑자기 사용되어 발생하는 '인지적 공백'을 파악하기 위해서다.
structural_ambiguity_detector	문법적으로는 맞지만, 수식 관계가 중의적인 문장(예: "귀여운 영수의	Knowledge Acquisition	독자가 문장을 읽다가 "어? 무슨 뜻이지?" 하고 다시 돌아가서 읽게 만드는 비효율적인 구간을 찾아낸다.

Tool 이름	기능	카테고리	활용 이유
	동생" - 영수가 귀여운지, 동생이 귀여운지)을 찾는다.		
cohesion_flow_tracer	문장과 문장 사이를 이어주는 연결 고리(접속사, 대명사, 지시어)가 자연스러운지 분석한다.	Knowledge Acquisition	논리적 비약이 있거나, '그러나' 같은 역접 접속사가 쓰였는데 내용이 반전되지 않는 등 독자의 예측을 빗나가게 하여 피로도를 높이는 구간을 식별한다.
semantic_consistency_checker	글을 따라 가면서 얻게 되는 단어의 의미와 쓰임이 자연스러운지 분석한다.	Knowledge Acquisition	비슷한 맥락이지만, 논리적 흐름보다는 용어의 의미에 중점을 둡니다. 글 전체에서 특정 용어가 동일한 정의로 사용되도록 보장합니다.
bridging_inference_checker	(가교 추론 검증) A문장에서 B문장으로 넘어갈 때, 독자가 스스로 채워 넣어야 하는 정보량이 너무 많은지 판단한다.	Knowledge Acquisition	글쓴이의 머릿속에만 있는 전제를 독자가 억지로 추론해야 하는 상황을 '불친절함'으로 규정하고, 이를 작성자에게 질문("이 사이에 생략된 전제가 있나요?") 형태로 되돌려준다.
vocabulary_extractor	…	Knowledge Acquisition	…
			타임라인 관리 툴 (1)
			…

추합된 에이전트와 그 툴:

에이전트 이름	역할 및 정의	소유한 도구
인지 및 논리 전문 에이전트	글의 논리적 흐름, 전제-결론의 타당성, 배경지식의 필요 여부를	knowledge_gap_scanner, bridging_inference_checker

	검증한다. 독자의 사고 과정에서 발생하는 인지적 장벽을 탐지한다.	
구조 및 구문 전문 에이전트	글의 문법적 구조나 수식 관계, 통사적인 문제 등을 분석한다. 형식이 내용을 정확하게 담고 있는지, 문장이 중의적으로 해석될 여지는 없는지 확인한다.	structural_ambiguity_detector
응집성 및 문맥 전문 에이전트	문장-문단 간의 연결이 자연스러운지, 동일한 개념이 일관된 용어로 사용되었는지를 더 큰 눈의 관점에서 추적한다.	cohesion_flow_tracer, semantic_consistency_checker,
최고 편집자 에이전트	전체 워크플로를 관리하는 최고 편집자 역할을 수행한다. 사용자 입력(원고 및 독자 페르소나)을 분석하고, 원고의 장르와 독자의 민감도에 따라 가장 적절한 분석을 수행하도록 전문 에이전트들에게 작업을 위임한다. 필요하다면 전 에이전트가 사용 가능한 사전 구축을 한다.	layout_context_preserver, 관리 성격이 있는 툴 (vocabulary_extractor 등)
피드백 통합 에이전트	각 전문 에이전트로부터 생성된 보고서를 수집하고, 최고 편집자의 가이드라인(민감도 및 우선순위)을 기준으로 필터링하며, 필요한 위치 및 표현 방법을 기록하고, 이 분석 결과를 종합하여 독자의 반응을 시각적 레이어(예: 밑줄, 메모)로 덧입힐 수 있도록 데이터를 JSON 형식으로 취합한다.	
…	…	…

[Step 2] 흐름 분석 및 상세 서술

Step 1에서 기획한 서비스가 실제로 어떻게 동작할지 텍스트로 풀어냅니다.

- **서비스 개요: 이 에이전트는 무엇을 해결하기 위한 것인가요?**

  글을 쓰는 사람은 자신의 의도를 이미 알고 있기 때문에, 자신의 글이 남들에게 어떻게 읽히는지 객관적으로 판단하기 어렵다. 이 서비스는 타인의 눈으로 내가 쓴 글을 보게 해준다. 독자가 겪을 오해나 피로감을 미리 시뮬레이션해 줌으로써 쓰는 이로 하여금 다시 생각해보는 데 도움을 준다.

- **이용자 관점 흐름: 사용자가 서비스를 시작해서 결과를 받을 때까지의 경험을 서술하세요. (Input → Output)**
  1. 원고 입력:
     - 사용자가 작성 중인 소설이나 기획서를 업로드한다.
  2. 예상 독자 페르소나 설정:
     - "이 글을 누가 읽나요?" (예: 해당 분야 전문가, 일반 대중, 혹은 꼼꼼한 편집자)를 선택하여 시뮬레이션의 기준(민감도)을 맞춘다.
  3. 시각적 피드백 확인:
     - 글이 변형되지 않고 그대로 출력되지만, 특정 구간에 예상 독자의 반응이 붙는다.
       - *예시:* "프로젝트 A"라는 단어에 노란 밑줄이 그어지고, 마우스를 올리면 *"이 용어는 3문단에서 처음 등장하는데, 앞서 설명된 적이 없어 낯설게 느껴집니다."* 라는 메시지가 뜬다.
       - *예시:* 복잡한 문장에 *"이 문장은 주어와 서술어 사이가 너무 멀어, 읽는 동안 무엇에 대한 이야기를 하는지 잊어버릴 위험이 있습니다."* 라는 코멘트가 달린다.
  4. 수정 및 반영:
     - 사용자는 지적된 부분을 수정하고, 실시간으로 해당 구간의 '읽기 걸림돌'이 사라지는 것을 확인한다.

- **시스템 관점 흐름: 내부적으로 데이터가 어떻게 흐르고, 어떤 Tool이 언제 호출되는지 논리적 순서를 서술하세요.**

  1. 초기 수집 및 분석 전략 수립

     - 사용자의 원고 + 설정된 독자 페르소나 (예: 비전공자, 전문가) + 글의 장르 (예: 소설, 기술 매뉴얼)가 시스템에 입력된다.
     - 최고 편집자 에이전트는 입력 데이터를 분석하여 텍스트를 구역별로 나누고, 동시에 분석의 '민감도 잣대' 및 '분석 우선순위'를 결정한다.
     - 문서를 분석하여 '제목', '본문', '인용구' 등을 구분한다.
     - 글의 길이가 길거나 기술 문서인 경우, 전체 문맥을 관통하는 용어 사전을 우선적으로 생성하도록 한다.
     -

2. 작업 위임

- 최고 편집자 에이전트는 문서의 골격(코드, 인용구, 제목 등)을 식별하도록 한 뒤, 그 정보를 바탕으로 나머지 전문 에이전트들에게 작업을 위임한다. 각 전문 에이전트에게 작업을 지시할 때 해당 구역의 속성(예: '이것은 코드 블록임') 정보를 함께 전달한다.
    - *예) 소설 등*: 인지 에이전트에게 특정 툴 사용을 비활성화하거나, 논리 오류 보고의 민감도 등을 완화하도록 지시한다. 이는 예술적 허용을 논리 오류로 오진하는 것을 방지하기 위함이다.
    - *예) 기술 매뉴얼 등*: 구조 에이전트에게 중의적인 내용이 있는지 확인한다. 동시에 응집성 에이전트에게는 용어 정의의 일관성(동일한 대상이 문맥에 따라 다른 용어로 지칭되어 혼란을 주지 않는가 등)을 최우선으로 검증하도록 명령한다.

3. 병렬 분석 및 보고서 작성

지시를 받은 각 전문 에이전트는 할당된 Tool을 사용하여 텍스트를 심층 분석하고, 발견된 이슈를 raw 형태의 보고서로 작성한다.

- 인지 에이전트: 전달받은 용어 리스트를 바탕으로, 각 단어의 정의가 최초 등장 시점보다 늦는지 검증한다(타임라인 관리 툴), 설명 없이 등장한 새 용어를 포착한다.
- 구조 에이전트: 해석의 가지가 갈라지는 중의적 문장이 없는지 등을 찾는다.
- 응집성 에이전트: 문장 간 의미 연결을 추적하여 흐름이 끊기거나 급격히 변화하는 구간을 찾는다. 동일한 대상이 문맥에 따라 다른 용어로 지칭되어 혼란을 주지 않는지 검증한다.
- ...

4. 종합 선정

- 피드백 통합 에이전트는 모든 전문 에이전트의 보고서를 취합한다.
- 단순히 탐지된 모든 것을 보여주면 사용자가 역으로 피로해진다. 탐지된 항목 중 상충되는 분석 결과가 있을 경우 독자 페르소나를 기준으로 우선순위를 판단하여 선별한다.
    - *예:* 독자 페르소나가 '전문가'인 경우, 기초 전문 용어 관련 이슈는 무시하거나 숨김 처리한다.
- 최종 이슈들만 UI에 표시될 수 있는 JSON 형식의 메타데이터(위치, 유형, 자연어 코멘트)로 변환하여 사용자에게 반환한다.
- 이 작업을 반복할 수도 있다.

- 구현 시 고민 사항: API 비용, 응답 속도, 환각(Hallucination) 방지 대책 등 예상되는 기술적 난관을 나열하세요.

1. 설계 미스로 그럴싸한 말이 만들어질 수 밖에 없을 수도

    - **문제:** LLM은 그럴듯한 논리 구조를 만들어내는 것을 쓸데없이 잘한다. 만약 에이전트가 원문에 존재하지 않는 인과관계를 있는 것처럼 그려주거나, 엉뚱한 구문 트리를 보여준다면 사용자는 자신의 글이 틀렸다고 오판하게 된다.

- **대책:** 한번 더 체크해서 전제와 결론이 정말인 경우에만 연결을 해준다. 확신할 수 없는 모호한 논리는 아예 시각화하지 않기로 한다.

2. 말하고자 하는 도메인에 관한 문제

- **문제:** 기술 문서의 엄격한 잣대를 감성적인 소설에 들이대면 재앙이 된다. 소설의 '극적인 긴장감'(현학적임)을 기술 문서의 '가독성 실패(높은 엔트로피)'로 오진하거나, 소설 속 '꿈 장면/회상'(논리가 없음, 생각의 흐름)같은 특수 맥락을 인식하지 못하고 논리 오류로 지적할 위험이 있다.
- **대책:** 분석 시작 전, 글의 장르에 따라 '놀람(Surprisal) 기준'을 동적으로 조정하는 단계를 둔다. 또한 시각적 구조(특수 태그 같은)를 통해 구역을 물리적으로 격리하여 분석한다.

3. '일부러 둔' 것들의 구분

- **문제:** 효율성만을 강조하다 보면, 독자의 이해를 돕기 위해 작성자가 의도적으로 넣은 요약이나 반복 강조를 불필요한 중복으로 표시할 수 있다.
- **대책:** 정보 밀도 분석 시, 문단이나 챕터의 시작과 끝부분에 나타나는 반복은 '논리 강화'로 간주하여 감점하지 않는 예외 로직을 적용한다?

4. 프롬프트 인젝션

- **문제:** 악의적이거나 의도치 않은 추가 지시가 몰래 삽입되어 전달될 경우, 심각한 혼란이나 시스템 오류가 발생할 수 있다.
- **대책:** 만약 추가적인 지시 사항을 문서에 포함시켜야 할 경우, 다음을 필요로 해야 할 것이다.
    a. **특별한 형식:** 지시를 담는 고유한 구조 또는 형식.
    b. **인식표와 검증기:** 해당 지시가 실행되어야 할 특정 상황에만 작동하는 식별 장치 및 검증 메커니즘.

또한 그 외 확장 시 발생가능한 다른 문제들(모델에 따른 컨텍스트 길이 등).

[Step 3] 도식화 (Diagramming)

글로 정리한 내용을 바탕으로 실제 개발이 가능하도록 두 가지 형태의 도면을 그립니다.

설계 #1 (기능 흐름도)

사용자는 '분석할 원고'와 '가상의 독자 페르소나(예: 비전공자, 꼼꼼한 편집자)'를 입력으로 제공한다. 시스템은 텍스트를 수정하지 않고, 원본 위에 인지적인 문제 등을 시각화한 '레이어'를 반환한다.

1. 입력:
    - 소설, 기술 문서, 기획서 등 텍스트 데이터.
    - 독자 페르소나(예: "배경지식이 없는 일반인", "업계 10년 차 전문가") 및 장르 지정.
2. 처리 과정:
    - 시스템은 먼저 글의 장르와 독자 수준을 파악하여 분석 전략을 수립.

    - 필요하면 어휘를 사전에 포함.
    - 이후 필요한 관점(논리, 구조, 흐름 등)만 선택적으로 문제를 탐지.
3. 출력 (예시):
    1. 구성: 원본 텍스트 위에 JSON 메타데이터 리스트 (클라이언트에서 유형별 색상 하이라이트로 렌더링)에 기반해서, 문제 유형별로 색상 하이라이트 혹은 문장 부호를 이용해서 적용.
    2. 노란색 (인지적 공백): "이 용어는 3문단에서 처음 등장하나 정의되지 않았습니다." (담당: 인지/논리 에이전트)
    3. 파란색 (구조적 모호성): "수식 관계가 중의적이라 독자가 두 번 읽게 만듭니다." (담당: 구조/구문 에이전트)
    4. 빨간색 (흐름 단절): "앞 문장(A)과 뒷 문장(B) 사이의 논리적 연결 고리가 부족합니다." (담당: 응집성/문맥 에이전트)
    5. 사이드 패널: 클릭 시 해당 에이전트가 분석한 상세 근거와 수정 제안 제공

설계 #2 (Agentic Workflow 관점)

이 워크플로는 최고 편집자가 하위 전문 에이전트들을 지휘하는 위계적 구조로 동작한다. 모든 툴을 무작정 실행하지 않고, 상황에 맞춰 동적으로 지시한다.

1. 최고 편집자의 감독 및 전략 수립

- 사용자의 입력(원고+페르소나)을 분석하여 작업 지시서를 시스템 프롬프트로 동적으로 생성.
- `layout_context_preserver`: 코드 블록, 인용구 등 특수 구조 식별.
- 글의 특성에 따라 어떤 전문 에이전트를 호출할지, 어떤 기준으로 분석할지 결정.
    - *예: 소설인 경우 '논리 에이전트'의 민감도를 낮추고, 기술 문서인 경우 '구조 에이전트'의 엄격함을 높임 .*
- 필요 시 사전을 생성하여 각 전문 에이전트의 프롬프트 컨텍스트에 주입.

2. 전문 에이전트 위임

‘최고 편집자’의 지시에 따라 3개의 전문 에이전트가 각자의 도구를 사용하여 심층 분석을 수행

예시:

- A. 인지 및 논리 전문가: 독자의 사고 과정을 시뮬레이션
    - `knowledge_gap_scanner`: 설명 없는 낯선 용어 탐지.
    - `bridging_inference_checker`: 문장 간 과도한 추론 요구(불친절함) 탐지.
    - ...
- B. 구조 및 구문 전문가: 텍스트의 형태와 문법적 명확성
    - `structural_ambiguity_detector`: 중의적 해석이 가능한 비효율적 구문 탐지.
    - …
- C. 응집성 및 문맥 전문가**:** 글의 흐름과 일관성
    - `cohesion_flow_tracer`: 접속사와 대명사의 자연스러운 연결 추적.
    - `semantic_consistency_checker`: 동일 개념의 용어 통일성 검증.

    - …
- Z. 그리고 다른 가능한 전문가 등..
    - …

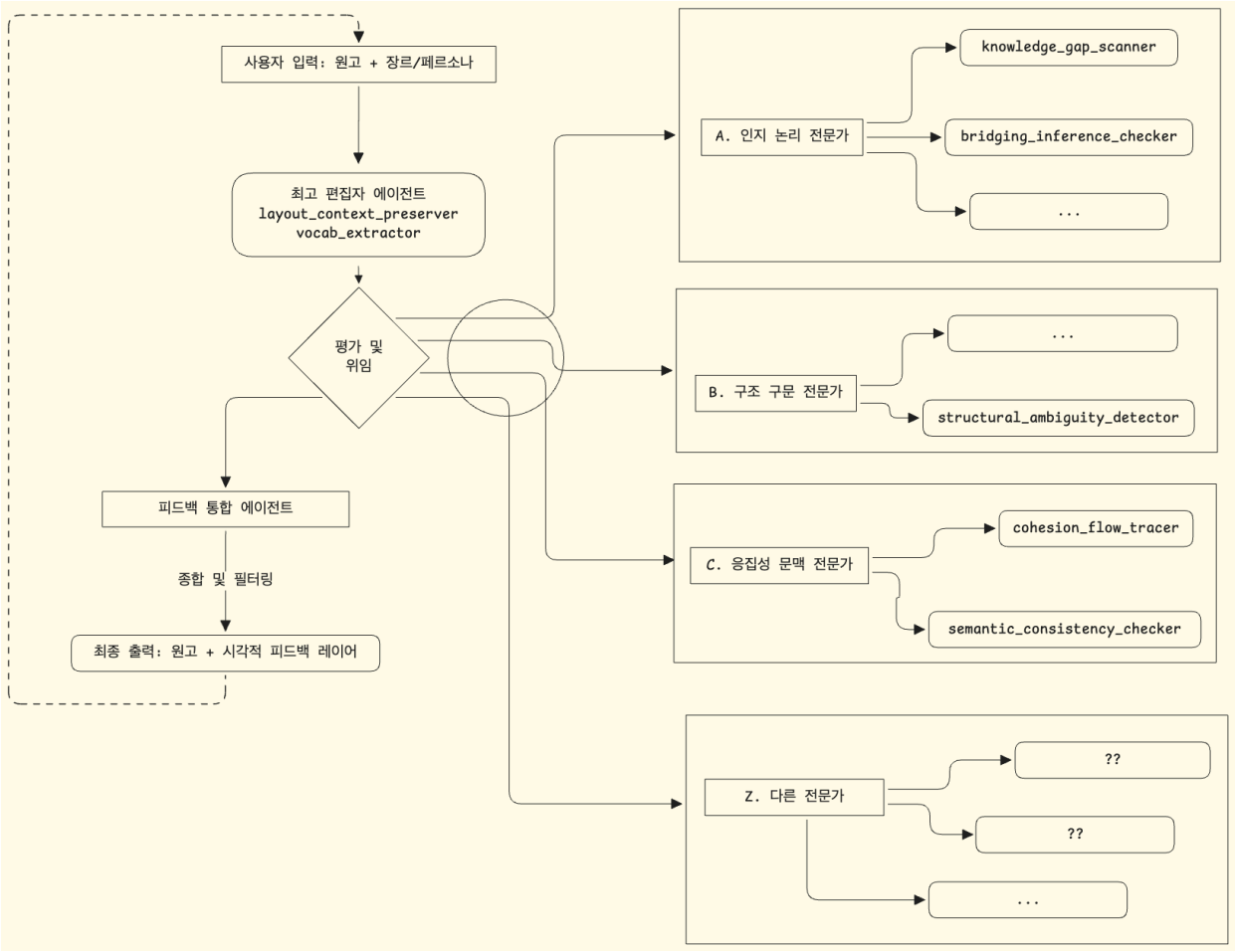
3. 종합

- 피드백 통합 에이전트가 각 전문가가 올린 날것 상태인 리포트를 취합, 독자 페르소나에 맞춰 기각하거나 승인함.

4. 결과 생성

- 최종 생존한 이슈들을 JSON 메타데이터(위치, 유형, 메시지 등)로 변환하여 UI로 전송.
- **만약 루프를 할 경우**: 부분 업데이트**.** 사용자가 특정 문단을 수정하면, 전체 프로세스를 다시 돌리지 않고 관련된 전문 에이전트만 호출하여 해당 구간의 레이어를 갱신.

다이어그램



예시 사진(극단적인 목표)

맨 처음에 만들어진 것이라서 툴 이름과 성격이 현재 상태와 맞지 않는 부분이 있을 수 있으나 대략적으로 이러한 모양이라고 보면 되겠음.

본 문서는 차세대 하이브리드-메시 아키텍처[101] 도입에 따라, 기존의 모놀리식 환경에서 발생했던 간헐적인 레이스 컨디션 문제를 해결하기 위해 작성되었으나, 사실상 3년 전 프로젝트 '게스팔'[102] 당시 제안되었던 낙관적 락(Optimistic Lock) 방식이 DB의 격리 수준을 'Repeatable Read'로 설정했을 때 발생하는 팬텀 리드 현상과 맞물려 의도치 않은 사이드 이펙트를 낼 수 있다는 점[103]이 발견됨에 따라, 이를[104] 전면 재검토하는 과정에서 분산 사가 패턴의 코레오그래피 방식[105]을 차용하기로 결정된 바, 이것[106]이 적용될 경우 이벤트 소싱 기반의 로그 리플레이가 필수적이라는 전제 하에, 만약 스냅샷 주기가 트랜잭션의 커밋 속도보다 느릴 경우 리하이드레이션 과정에서 OOM이 발생할 수 있는데[107], 이는 결과적으로 카프카의 파티션 리밸런싱을 유발하여 전체 컨슈머 그룹의 랙을 증가시키는 치명적인 안티 패턴이 될 수도 있고 아닐 수도 있다는 의견이 분분하여[108] (사실 이 부분은 당시 김 수석님이 강력하게 주장했던 부분이라 어쩔 수 없이 들어간 코드가 있긴 함), 우리는 서킷 브레이커[109]의 임계치를 동적으로 조절하는 어댑티브 스로틀링[110]을 도입하여 이 문제[111]를 우회하려고 시도할 예정이다.

또한, 레거시 모듈인 `LegacyOrderProcessor` 클래스 내부의 `execute()` 메서드[112]는 동기식으로 호출되도록 설계되어 있어, 외부 결제 API의 응답이 지연될 경우 워커 스레드가 블로킹되는 현상이 관측되었으며, 이러한 구조적 한계로 인해 톰캣의 스레드 풀이 고갈되는 현상이 블랙 프라이데이 트래픽 상황에서 재현될 가능성이 농후하므로[113], 비동기 Non-blocking I/O[114] 기반의 웹플럭스 전환이 시급하지만, 기존의 JPA가 블로킹 드라이버를 사용한다는 점에서 완전한 리액티브 스택으로의 전환은 R2DBC의 성숙도를 고려할 때 시기상조라는 판단 하에[115], 일단은 `@Async` 어노테이션을 활용하여 별도의 ExcecutorService로 격리하는 방안이 채택되었으나, 그럴 경우[116] 트랜잭션 컨텍스트가 전파되지 않는 문제가 발생하여 롤백이 제대로 수행되지 않을 위험이 존재한다는 사실을 간과해서는 안 될 것이다.

따라서, 2PC[117]의 대안으로 제시된 TCC 패턴을 적용함에 있어, Try 단계에서 리소스를 예약하는 과정이 레디스의 Lua Script를 통해 원자적으로 수행되어야 함은 자명하나, 만약 레디스 클러스터의 마스터 노드가 페일오버되는 시점과 스크립트 실행 시점이 겹칠 경우 데이터 정합성이 깨질 수 있는 엣지 케이스에 대한 대비책이 현재로서는 전무하며[118], 이를 해결하기 위해 Redlock 알고리즘을 도입하자는 의견도 있었으나 네트워크 파티션 상황에서 CAP 정리에 의해 가용성을 희생해야 한다는 점[119]이 비즈니스 요구사항(무중단 서비스)과 정면으로 배치되기 때문에, 결국 결과적 일관성을 수용하되, 별도의 배치 프로그램을 통해 주기적으로 데이터 불일치를 보정하는 방식[120]으로 타협점을 찾았으나, 이 배치가 도는 시간에 DB 부하가 급증하여 정작 실시간 서비스가 느려지는 주객전도 현상[121]이 발생하고 있는 실정이다. 이에 우리는 DB 부하를 원천 차단하기 위해 배치 프로그램을 제거하고, 다시 초기 제안이었던 낙관적 락 방식을 재검토하기로 하였다.[122]

[예시]

- **첫 번째 문단은 단 하나의 문장(532자)으로 구성되어 있으며, 내포절의 깊이가 4단계에 달합니다.** 이는 보편적인 인간의 작업 기억 용량을 아득히 넘습니다.
- **정보 불균형:** [108] "사실 이 부분은 당시 김 수석님이 강력하게 주장했던..." 구간은 기술적 인과관계가 없습니다. 핵심 개념이 아닌 부가 설명에 정보량이 집중되어 독해 속도를 저하시킵니다.

knowledge_gap_scanner 탐지 항목

- **[101] 차세대 하이브리드-메시 아키텍처:** 문서 내에서 처음 등장하는 고유 명사이나, 해당 용어에 대한 정의가 선행되지 않았습니다. 일반적인 'Service Mesh'와 혼동될 여지가 있으며, 독자는 이것이 업계 표준 용어인지 사내 프로젝트명인지 판단할 수 없습니다.
- **[102] 프로젝트 '게스팔':** '게스팔'라는 대명사는 특정 컨텍스트(3년 전, 사내)를 공유하는 소수의 인원만 해독 가능합니다. 일반적인 시니어 엔지니어에게는 불투명한 정보입니다.
- **[112] LegacyOrderProcessor / [113] 블랙 프라이데이:** 아키텍처 레벨(High-level)의 논의 중 갑자기 특정 클래스 메서드(Low-level)와 특정 마케팅 이벤트 시점이 등장합니다. 추상화 수준이 급격히 변동하여 문맥 파악을 방해합니다.

bridging_inference_checker 및 structural_ambiguity_detector 탐지 항목

- **[104], [106], [111], [116] 지시 대명사의 표류:**
  - **[104] "이를":** '낙관적 락'을 의미하는지, '팬텀 리드 현상'을 의미하는지, '사이드 이펙트'를 의미하는지 구문 상 모호합니다.
  - **[106] "이것":** '분산 사가 패턴'인지, '코레오그래피 방식'인지, 아니면 '전면 재검토 과정' 자체인지 불분명합니다.
  - **[116] "그럴 경우":** @Async를 사용했을 때를 말하는지, ExecutorService로 격리했을 때를 말하는지 명확하지 않습니다.
- **[103] ~ [107] 의존 거리:** 주어와 서술어, 혹은 조건과 결과 사이의 거리가 너무 멉니다. [107]의 "만약 스냅샷 주기가... 느릴 경우"라는 조건절은 [103]의 "사이드 이펙트를 낼 수 있다는 점"이라는 먼 과거의 문맥 속에 내포된 또 다른 하위 절입니다.

bridging_inference_checker 탐지 항목

- **[109] 서킷 브레이커 & [110] 어댑티브 스로틀링:** 앞선 문맥은 'DB 트랜잭션'과 '카프카 랙'에 대한 문제였습니다. 갑자기 '서킷 브레이커'라는 해결책이 등장하려면, 그 사이에 "동기 호출의 실패가 시스템 전반으로 전파되는 것을 막기 위해"와 같은 가교 역할 문장이 필요합니다. 현재 문맥에서는 인과관계가 단절되어 있습니다.
- **[115] R2DBC 성숙도 판단:** "R2DBC의 성숙도를 고려할 때 시기상조"라는 판단은 작성자의 주관입니다. 독자가 이 논리에 동의하려면 'R2DBC의 구체적인 결함 사례'나 '벤치마크 데이터'가 제시되어야 합니다.
- **[119] CAP 정리와 가용성 희생:** Redlock 알고리즘을 설명하기 위해 CAP 정리까지 끌어들이는 것은 설명의 경제성을 해칩니다. 비즈니스 요구사항(무중단)과 충돌한다는 결론만으로 충분함에도, 불필요한 이론적 배경을 나열하여 논점 일탈을 유발합니다.
- **[122] 논리적 회귀:** 문서 초반 [103]에서 '낙관적 락'은 '팬텀 리드' 및 '사이드 이펙트'로 인해 명시적으로 기각되었습니다. 그러나 문서의 결론 [122]에서 '배치 부하'를 이유로 다시 '낙관적 락'을 도입한다고 선언했습니다. 이는 "[103]의 문제(팬텀 리드)가 [122] 시점에서 해결되었는가?"에 대한 설명이 누락된 상태에서 회귀입니다.