Multi-Agent System for Automated Presentation Creation

Agenda

1. Introduction
2. System Overview
3. Architecture Style & Patterns
4. Agent Design Patterns (Anthropic-based)
    - Orchestrator-Worker
    - Evaluator-Optimizer
    - Prompt Chaining
5. Use Cases
6. Why We Chose n8n
7. Limitations and Challenges

1. Introduction

- **Objective:** Automate the creation of high-quality presentations from academic or business documents.
- **Driven by:** Large Language Models (LLMs), vector databases, and multi-agent coordination.
- **Motivation:** Enhance productivity in knowledge work by reducing manual effort and improving output quality.

2. System Overview

- **Trigger:** User uploads PDF to Google Drive
- **Processing Agents:**
  - Importer → Extracts text/images
  - Planner → Outlines presentation
  - Author → Creates slides in Markdown
  - Evaluator → Reviews and scores content
  - Feedback Agent → Refines based on feedback
- **Tools Used:**
  - PyMuPDF, PostgreSQL + PGVector
  - OpenAI/OpenRouter APIs

3. Architecture Style & Patterns

- **Architecture Style:**
  - Modular Multi-Agent System
  - Microservice Composition
  - Event-Driven Execution
  - Dataflow-Centric Design
- **Design Patterns Applied:**
  - **Blackboard Pattern** – Shared database for task handoff
  - **Chain-of-Responsibility** – Sequential task processing
  - **Evaluator Feedback Loop** – Iterative improvement via agent review

4. Anthropic-Inspired Agent Patterns

4.1 Orchestrator-Worker Pattern

- Central orchestrator (n8n) assigns tasks dynamically to worker agents.
- Workers (e.g., Author, Evaluator) are stateless but deterministic.
- Useful for varied slide creation and conditional agent branching.

4.2 Evaluator-Optimizer Pattern

- Slide Author produces content → Evaluator critiques it → Feedback applied.

5. Use Cases

UC1: Upload Paper

- User uploads PDF → n8n triggers workflow via Google Drive event.

UC2: Extract Text Content

- Text and images are extracted → Stored in vector DB for embedding-based search.

UC3: Plan Slides

- Planner Agent generates structured outline → JSON format.

6. Why We Use n8n

Pros:

- **Visual Workflow Design**: Easy orchestration of agent chains.
- **Native Integrations**: Google Drive, OpenAI, HTTP, databases.
- **Retry, Branching, Error Paths**: Orchestrator behavior out-of-the-box.
- **Hybrid Execution**: Combine visual flow with Python/JS scripts.
- **Persistence**: Workflow state can be saved for long-running tasks.

Ideal For:

7. Limitations and Challenges

Systemic:

- No real agent autonomy or collaboration
- Each agent works in isolation, lacks shared memory

Technical:

- n8n has no native long-term memory
- Large PDF files → LLM token limits (~128k max context)
- No built-in OCR or structured diagram parsing

Accuracy:

8. Summary

- Our multi-agent system automates end-to-end presentation creation using LLMs.
- The architecture follows proven patterns: **Prompt Chaining**, **Orchestration**, and **Evaluator Loops**.
- **n8n** enables low-code orchestration and robust API integration.
- While powerful, the system has technical and architectural limits that must be managed for production-grade usage.

Thank You

Questions? Discussion?