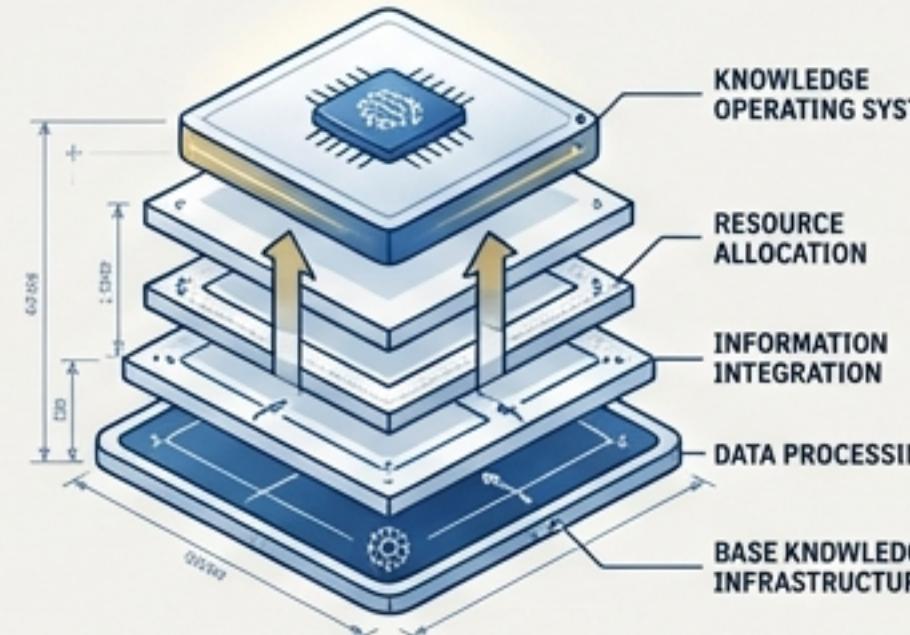
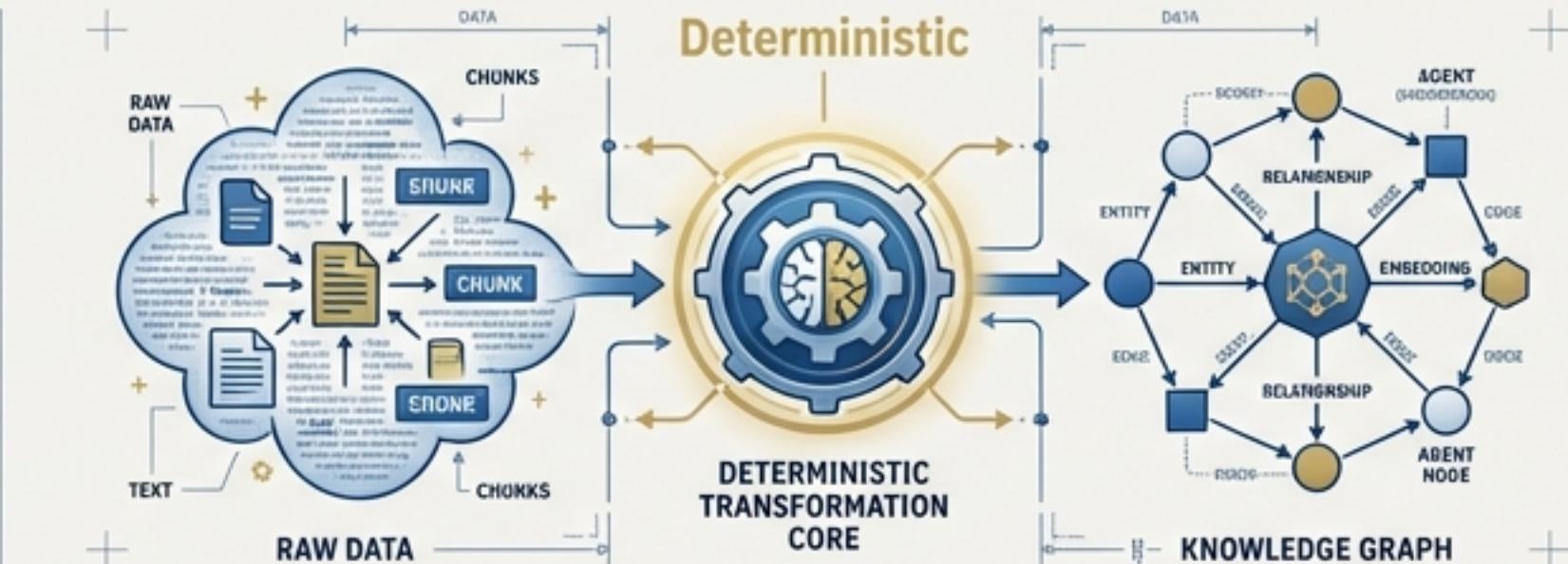


# UET Platform: The Knowledge Operating System

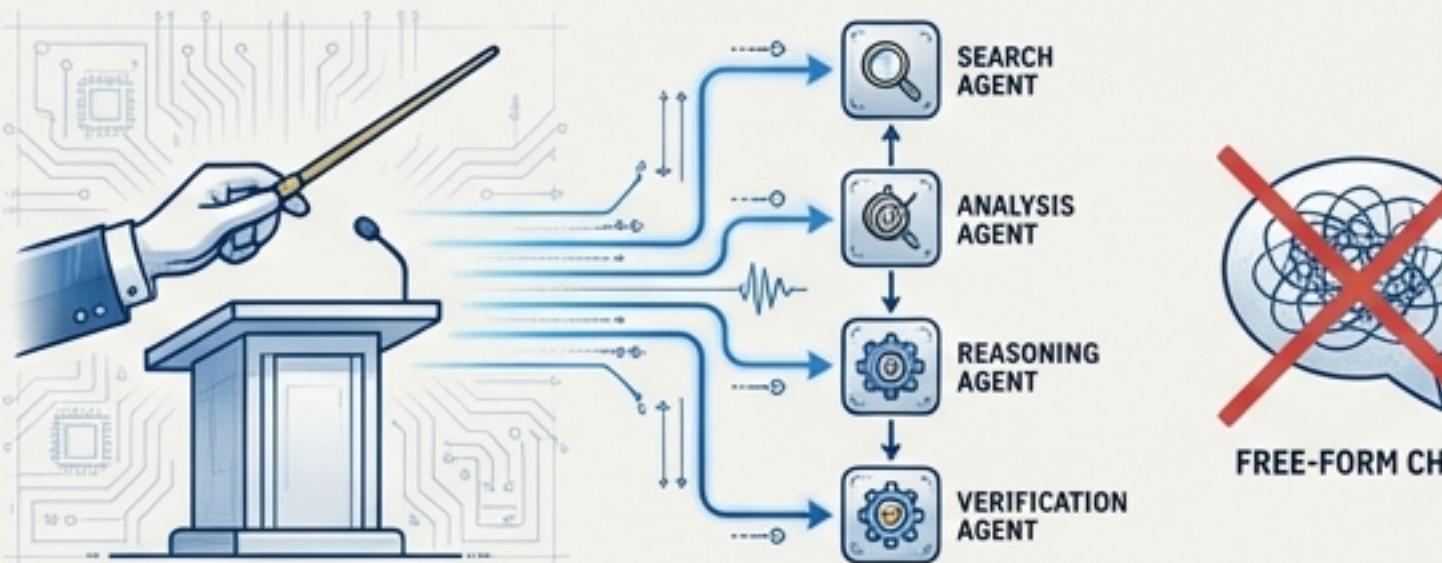
## นี่ไม่ใช่ LLM Chat แต่นี่คือระบบปฏิบัติการสำหรับองค์ความรู้



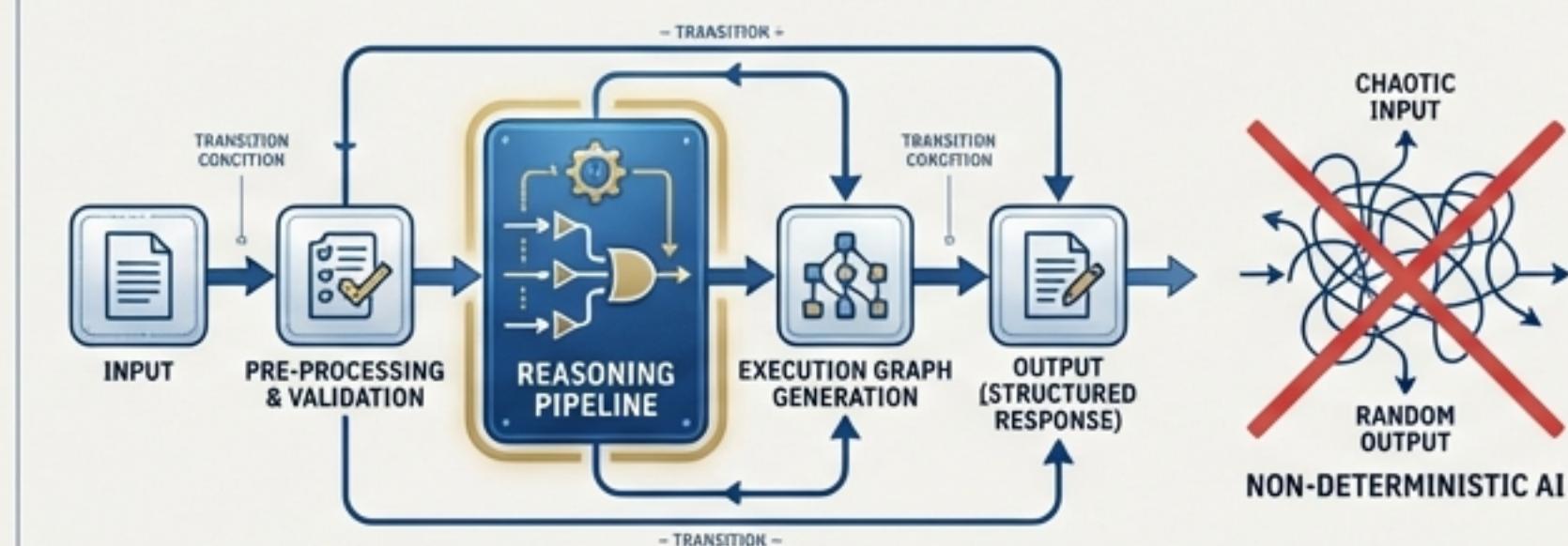
1. **Knowledge Operating System**: ทำหน้าที่เป็นตัวกลางในการจัดการองค์ความรู้ทั้งหมด ของผู้ใช้ ไปยังแค่การถาม-ตอบ



2. **Precision-RAG + Unified Knowledge Graph**: ศึกษาที่มีสาม Text, Chunk, Embedding, Graph, และ Agent เข้าด้วยกันอย่างเป็นระบบ (Deterministic) เพื่อสร้างคำตอบที่แน่นหนา



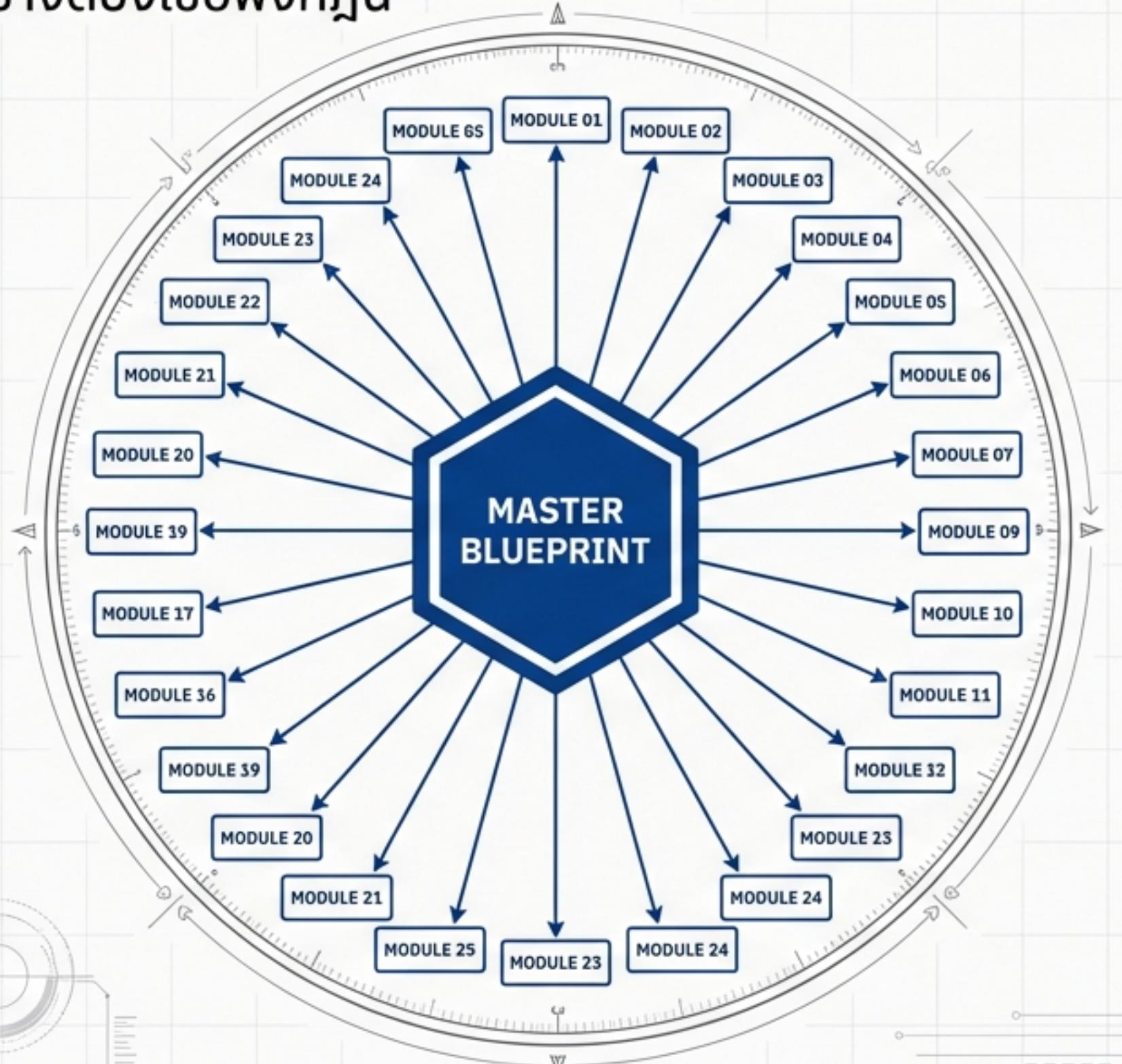
3. **Agent Orchestration System**: เป็นชั้นที่ UET Platform นำไปใช้ AI ที่ลึกๆ ไปเรื่อยๆ แต่เป็นระบบ "Orchestration" ที่ควบคุมการทำงานของ Agent ทุกขั้นตอน



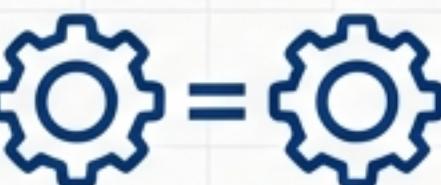
4. **Execution-Graph AI**: สร้างความแนบทẽต่ำงด้วยการระบุว่า AI ของเรามีไฉไล 'ศักดิ์สิทธิ์' และทำงานตาม Pipeline ทางไนโตรเจน (Reasoning) ที่มี State และกฎเกณฑ์ชัดเจนที่สุดExecution Graph

# รากฐานของความน่าเชื่อถือ: สัญญาสูงสุดของระบบ (The System Contract)

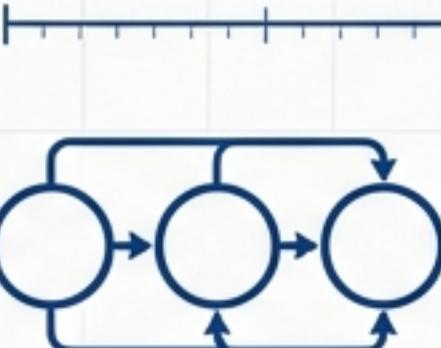
## ทุกอย่างต้องเชื่อฟังกฎนี้



**Blueprint คือศูนย์กลาง:**  
ทุกการอัปเดตต้องเริ่มต้นจาก  
Master Blueprint และกระจาย  
(Cascade) ไปยังโมดูลอยู่ 25 ไฟล์  
ห้ามมีการแก้ไขโดยไม่ว่างอิงตัวแม่



**ระบบต้องเป็น Deterministic:**  
เมื่อให้ Input ที่เหมือนกัน ต้องได้  
ผลลัพธ์ที่เหมือนกันเสมอ 100%  
นี่คือหัวใจที่ทำให้ระบบ 'ไว้ใจได้'



**ไม่มีการเป็นเจ้าของระบบ:**  
ไม่ถูกครอบครองแต่ละตัวทำงานตามหน้าที่  
ของตัวเองภายใต้พิมพ์เขียวหลัก  
ไม่มีการทำงานที่ขัดแย้งกัน

# สถาปัตยกรรมแห่งปัญญา: 6 ชั้นของการสกัดองค์ความรู้ (Knowledge Abstraction Layers)

**L5: Reasoning Blocks** - สมองของระบบ  
(Argument Structure, Synthesis)

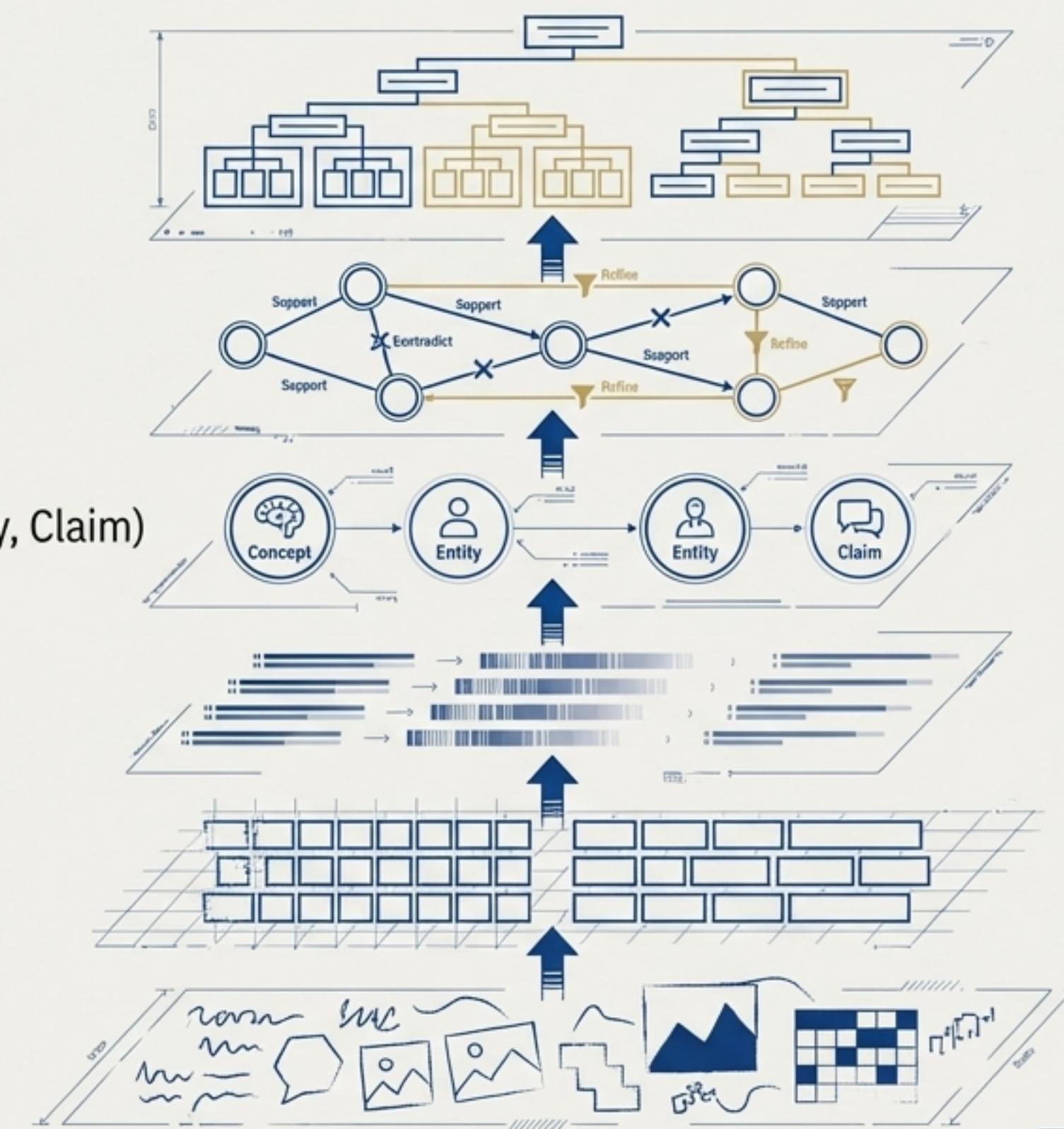
**L4: Relation Graph** - การสร้างความเชื่อมโยงเชิงเหตุผล  
(Support, Contradict, Refine)

**L3: Semantic Node** - การสกัด ‘ความหมาย’ (Concept, Entity, Claim)

**L2: Embedding** - การแปลง Chunk เป็น Vector ตัวเลข

**L1: Chunk** - การแบ่งข้อมูลเป็นหน่วยย่อย

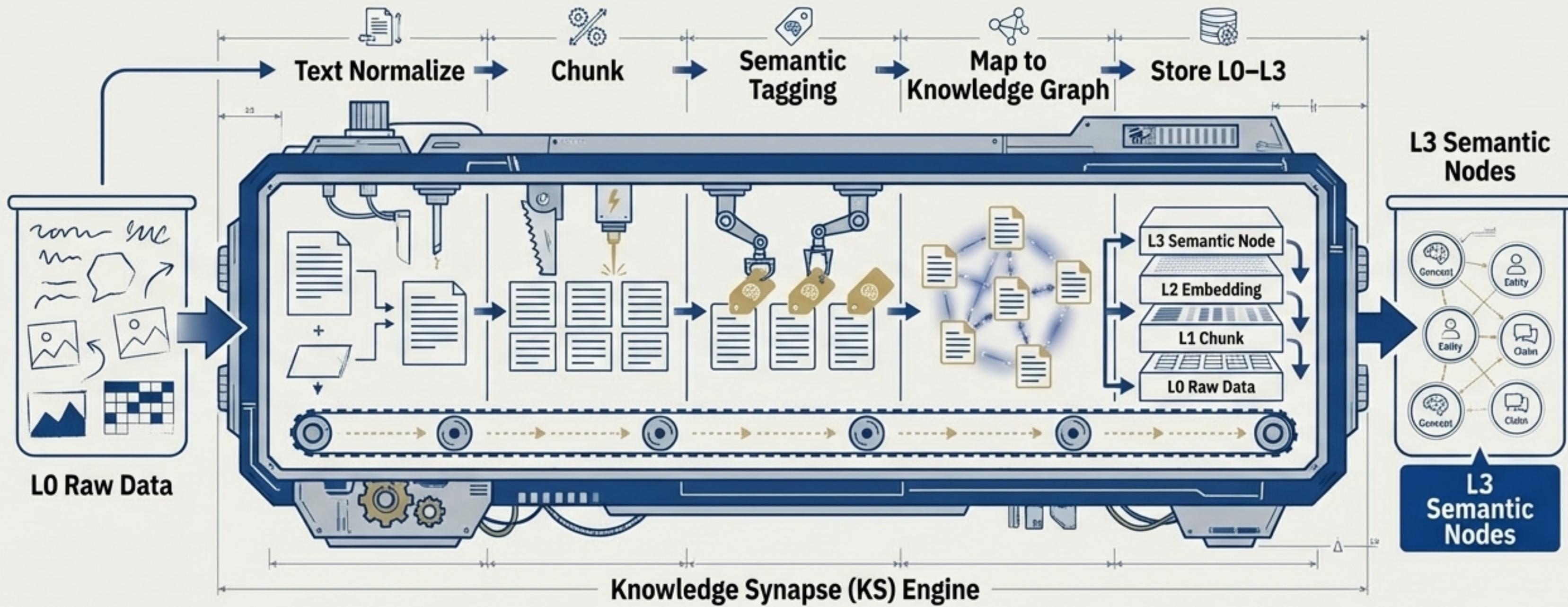
**L0: Raw Data** - ข้อมูลดิบ (Text, Image, Table)



ENGINEERING DOCUMENT

# KS Engine: เครื่องจักรเปลี่ยนข้อมูลดิบสู่องค์ความรู้ที่มีโครงสร้าง ( $L0 \rightarrow L3$ )

หน้าที่หลัก: รับข้อมูลดิบเข้ามาและแปลงให้เป็น 'Semantic Nodes' ( $L3$ )  
ซึ่งเป็นหน่วยความรู้พื้นฐานที่ระบบจะนำไปใช้ต่อ



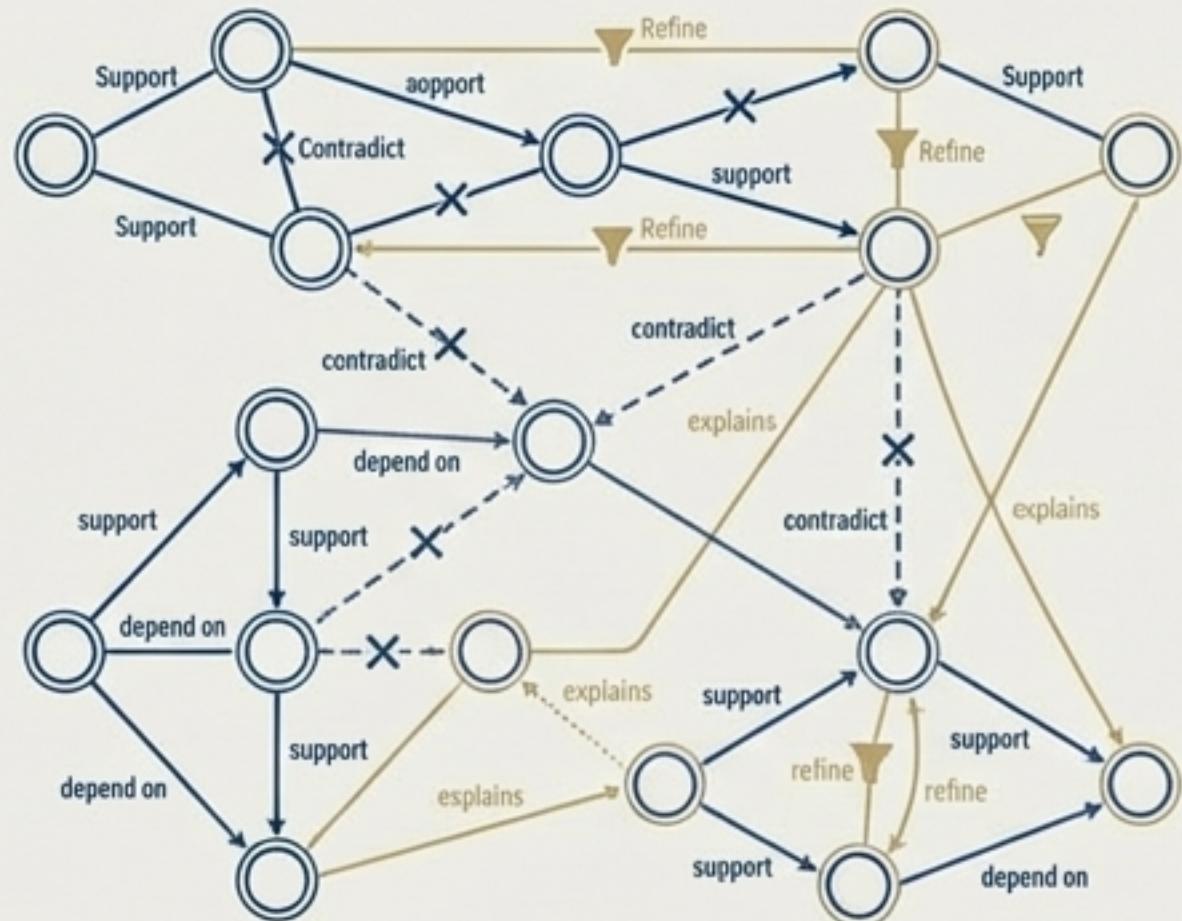
Knowledge Synapse (KS) Engine



ENGINEERING DOCUMENT

# Agent Engine & Execution Graph: สมองและการตัดสินใจของระบบ (L4 → L5)

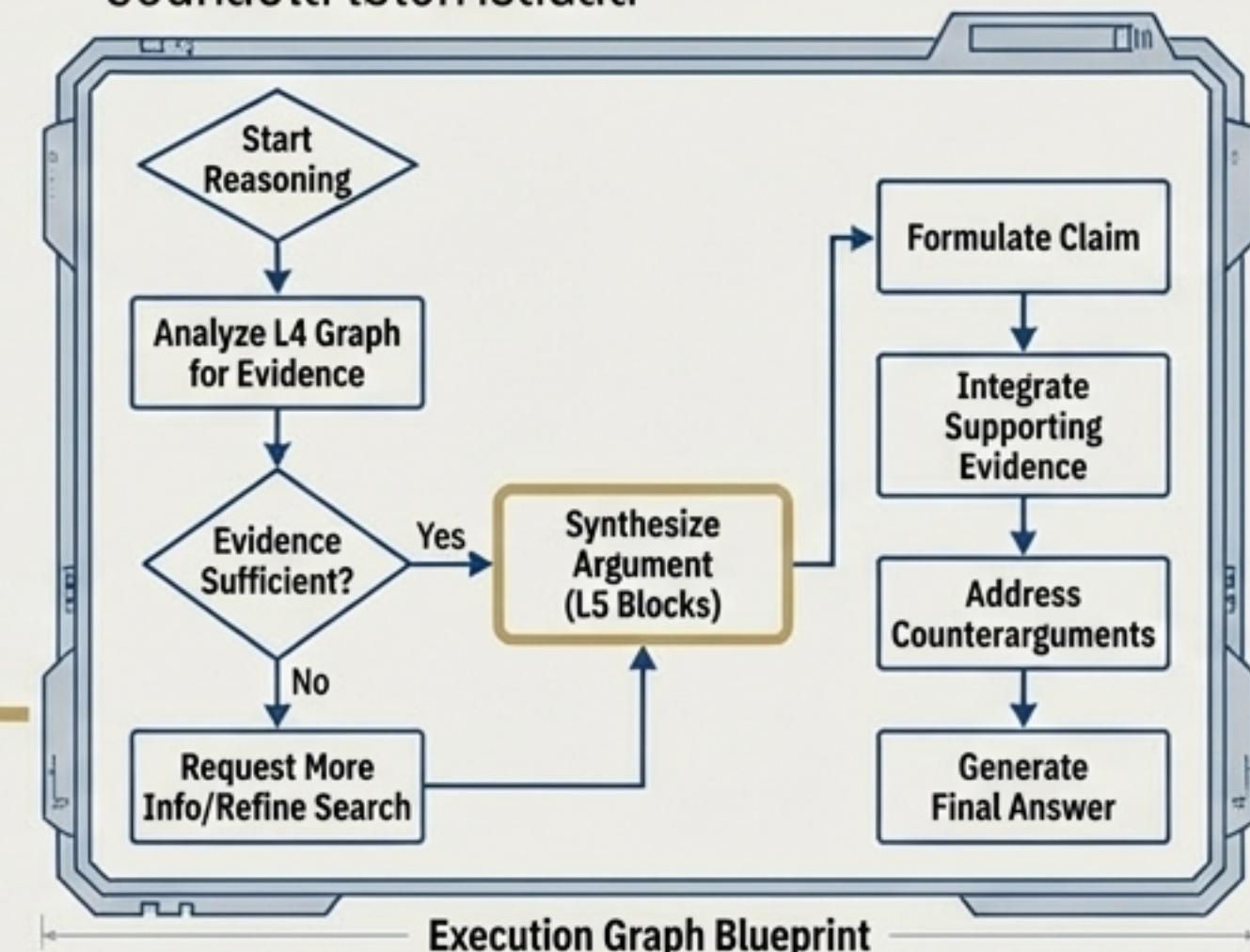
**L4 - Relation Graph:** แสดงให้เห็นว่าระบบไม่ได้แค่เก็บข้อมูล แต่สร้าง ‘เว็บแห่งความรู้’ (Knowledge Web) ที่เชื่อมโยงกันด้วยความสัมพันธ์เชิงเหตุผล เช่น ‘support’, ‘contradict’, ‘depend on’ ทำให้การให้เหตุผลมีความลึกซึ้ง



**Agent Engine:** ทำหน้าที่เป็น ‘วงเยก’ (Orchestrator) ที่นำองค์ความรู้จาก L4 และ L5 มาสังเคราะห์และสร้างเป็นคำตอบสุดท้ายตาม Execution Graph

**L5 - Reasoning Blocks & Execution Graph:**

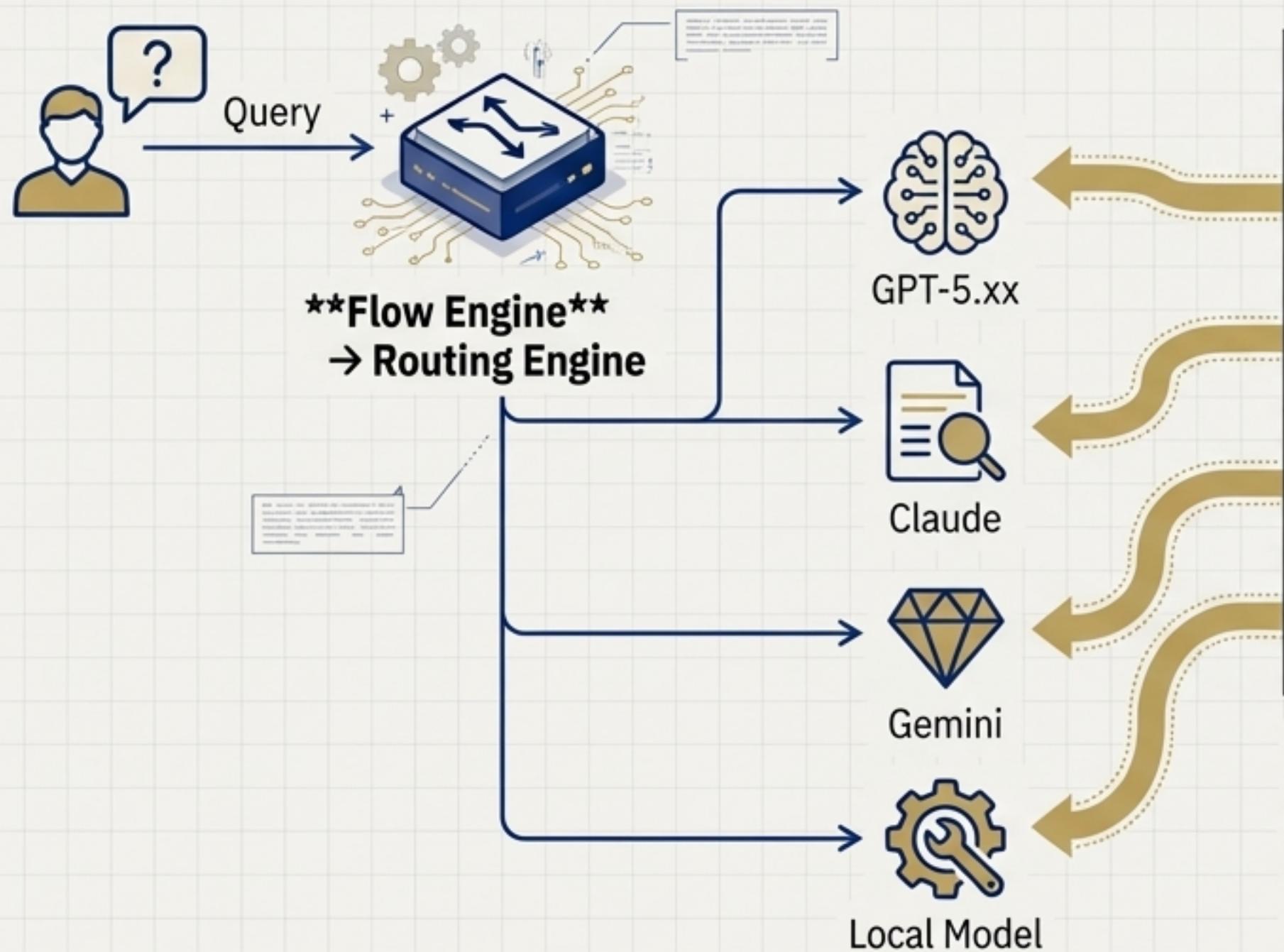
- บีคือ ‘สนอง’ ที่แท้จริงของระบบ ประกอบด้วยโครงสร้างการอ้างเหตุผล (Argument Structure) และรูปแบบการสังเคราะห์ข้อมูล (Synthesis Block)
- Execution Graph** คือแผนผังการทำงานที่กำหนดไว้ล่วงหน้า ทำให้ AI ‘คิด’ อย่างเป็นขั้นตอนและตรวจสอบย้อนกลับได้ ไปใช้การดันสตด



ENGINEERING DOCUMENT

# การเดินทางของคำสั่ง (End-to-End Flow): จากคำสั่งสู่คำตอบ

Step 1: เมื่อคำสั่งมาถึง... Routing Engine คือด้านแรก



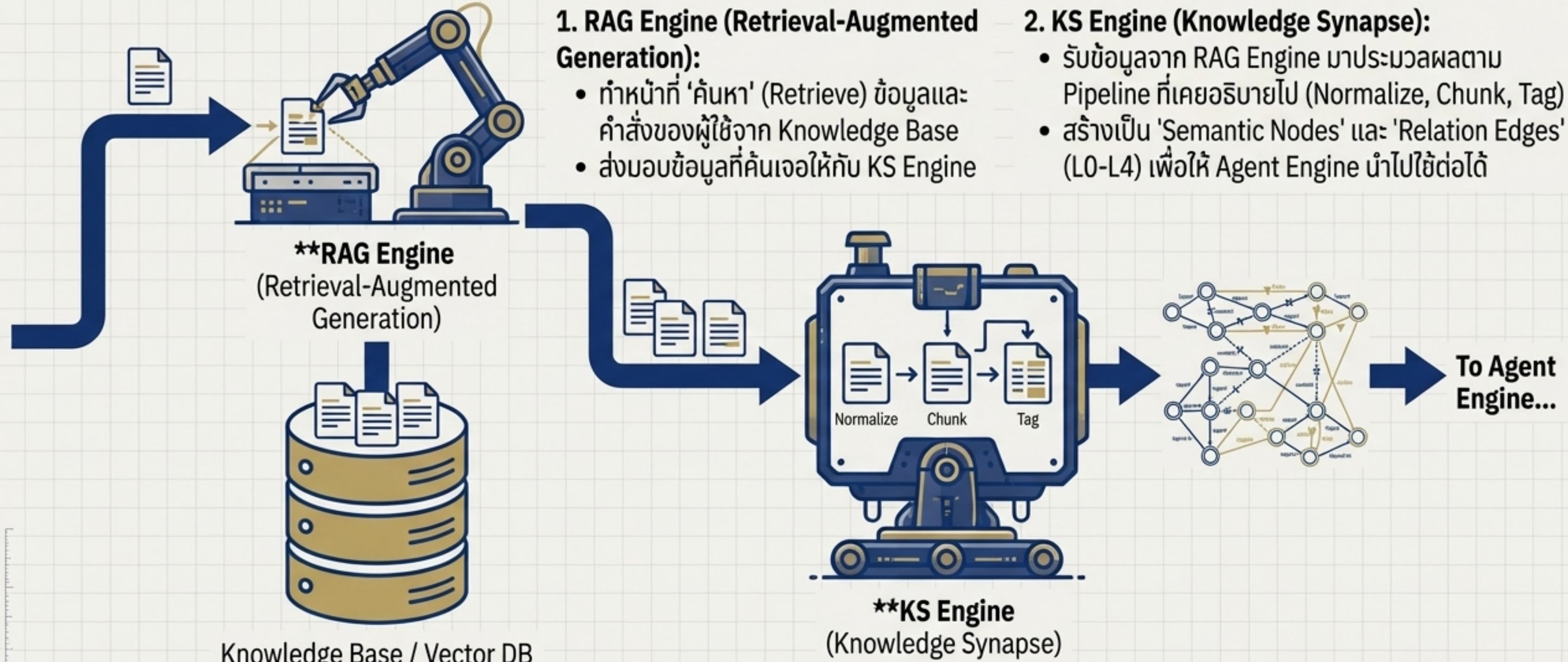
เกณฑ์การตัดสินใจ (Task Classification) → โมเดลที่เหมาะสม	
งานที่ต้องใช้เหตุผลหนัก (Reasoning-heavy)	ใช้โมเดลที่โหลดที่สุด (เช่น GPT-5.xx)
งานที่เน้นการดึงข้อมูล (RAG-heavy)	ใช้โมเดลที่เก่งด้านสรุปความ (เช่น Claude)
งานสรุปผลก้าวไป (Summarization)	ใช้โมเดลที่คุ้มค่า (เช่น Gemini)
งานเรียกใช้เครื่องมือ (Tool-call)	อาจใช้ Local Model เพื่อความเร็ว

**Flow Engine** รับคำสั่งและส่งต่อให้ **Routing Engine**  
**Routing Engine** กำหนดภาระที่จะประมวลผล (Task Classification) เพื่อเลือกใช้โมเดล (LLM) ที่เหมาะสมที่สุด สำหรับงานนั้นๆ



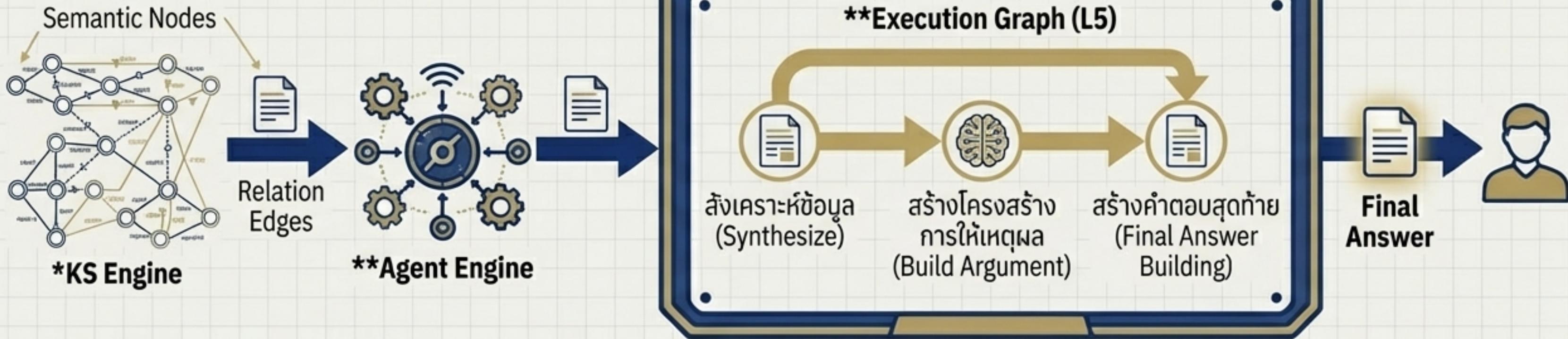
ENGINEERING DOCUMENT

# การเดินทางของคำสั่ง: Step 2: การค้นหาและสกัดองค์ความรู้



ENGINEERING DOCUMENT

# การเดินทางของคำสั่ง: Step 3: สังเคราะห์คำตอบผ่าน Execution Graph

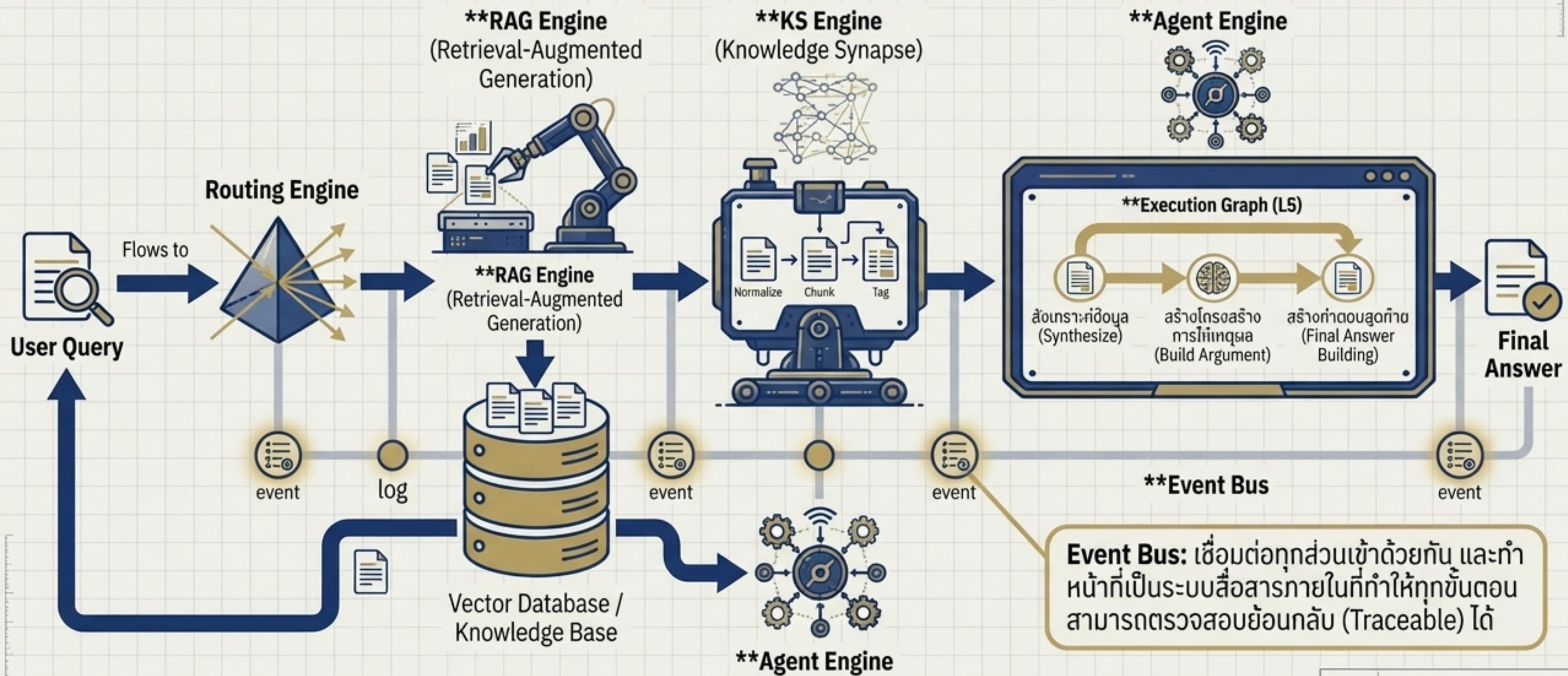


- **Agent Engine** รับข้อมูลที่ผ่านการประมวลผลจาก KS Engine
- นำข้อมูลนั้นเข้าสู่ **Execution Graph (L5)** ซึ่งเป็นเหมือน 'สูตร' ในการสร้างคำตอบ
- Agent จะดำเนินตามขั้นตอนใน Graph เช่น:
  - สังเคราะห์ข้อมูล (Synthesize)
  - สร้างโครงสร้างการให้เหตุผล (Build Argument)
  - สร้างคำตอบสุดท้าย (Final Answer Building)
- ส่งคำตอบที่สมบูรณ์และน่าเชื่อถือกลับไปยังผู้ใช้



ENGINEERING DOCUMENT

# ภาพรวมการทำงานแบบ End-to-End



ENGINEERING DOCUMENT

# เหตุผลเบื้องหลังการออกแบบ: ทำไมต้อง Deterministic และ Trustworthy?



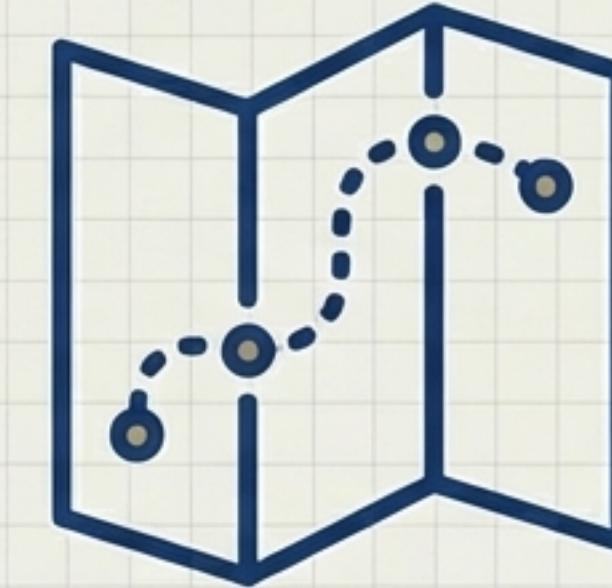
## ทำไมต้อง Deterministic?

- เพื่อให้ได้ผลลัพธ์ที่สม่ำเสมอและคาดเดาได้
- ลดปัญหา “AI หลอน” (Hallucination) หรือการตอบผิด
- สำคัญอย่างยิ่งในงานที่ต้องการความแม่นยำสูง



## อะไรที่ทำให้ระบบ ‘ไว้ใจได้’?

- System Contract:** กฎเหล็กที่ทุกคนต้องทำตาม ทำให้ระบบมีระเบียบ
- Execution Graph:** การทำงานเป็นไปตามแผน ไม่ใช่การดันสอด ทำให้ตรวจสอบได้



## อะไรที่ทำให้ระบบ ‘ไว้ใจได้’? (ต่อ)

- Event Bus System:** ทุกเหตุการณ์ในระบบถูกบันทึกและติดตามได้ (Log + Trace) ทำให้รู้ว่าเกิดอะไรขึ้น ที่ไหน เมื่อไหร่
- Traceability:** สามารถตรวจสอบย้อนกลับได้ทุกเส้นทาง

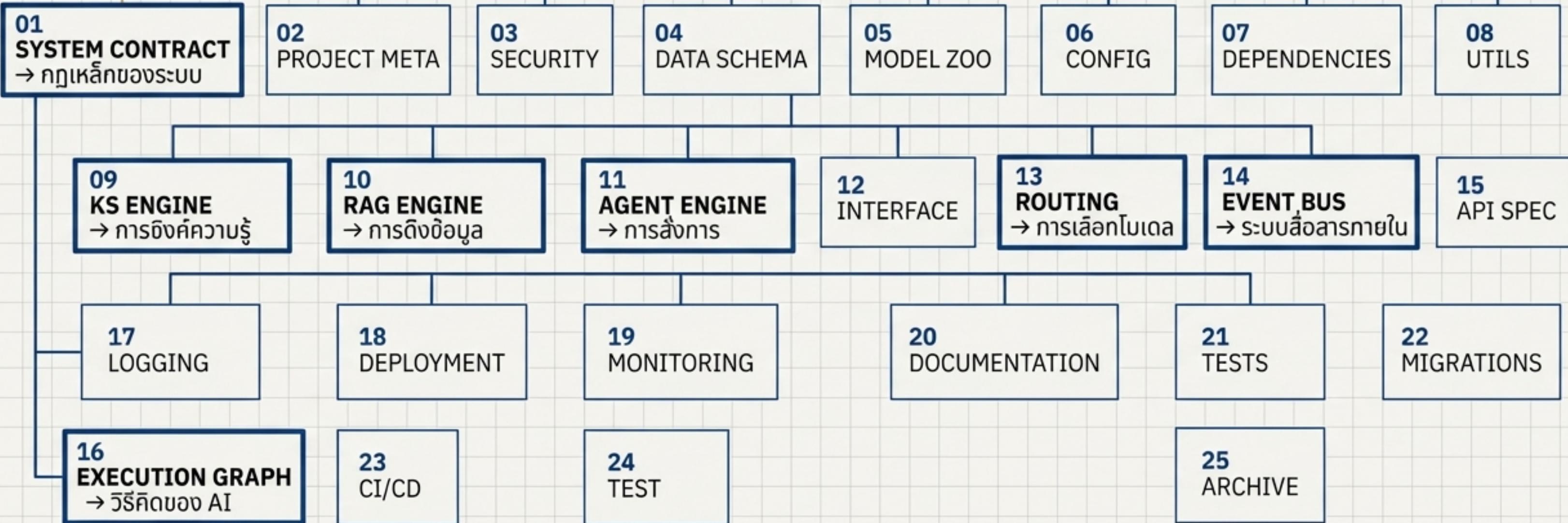


ENGINEERING DOCUMENT

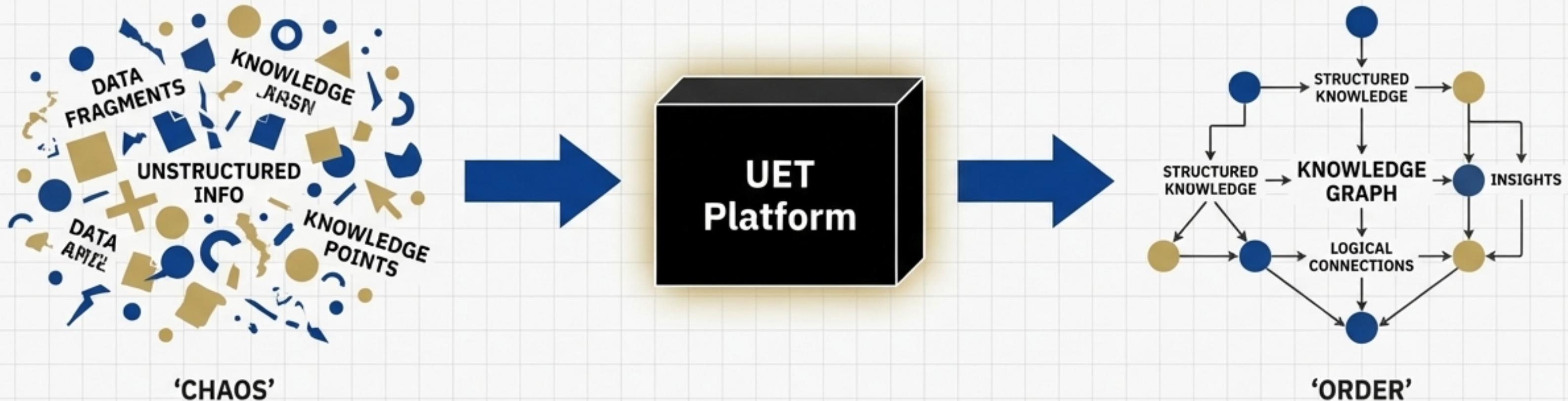
# The Complete Blueprint: ทุกองค์ประกอบถูกนิยามไว้

## MASTER\_BLUEPRINT

Blueprint ตัวตั้ง: โครงสร้างพื้นฐานที่สำคัญ



# สรุปง่ายๆ: UET Platform คืออะไรกันแน่?



โอเคครับ... สรุปแล้ว UET Platform มันคืออะไร? ถ้าให้พูดง่ายที่สุด มันไม่ใช่แค่ AI ที่เอาไว้คุยกันแล้ว แต่ มันคือ 'ระบบปฏิบัติการสำหรับความรู้' ครับ ลองนึกภาพว่าเรามีข้อมูลกระฉัดกระจายเต็มไปหมด แพลตฟอร์มนี้จะทำหน้าที่เหมือน 'สมอง' ที่คอยจัดระเบียบข้อมูลเหล่านั้น เปลี่ยนมันให้กลายเป็น 'ความรู้' ที่เชื่อมโยงกันเป็นเหตุเป็นผล เวลาเราถามอะไรเข้าไป มันไปได้ 'เดา' คำตอบ แต่ มันจะวิ่งไปตาม 'แผนที่' ที่เราวางไว้ (นั่นคือ Execution Graph) มันจะรู้ว่าต้องไปหยิบข้อมูลจากตรงไหน (RAG Engine) ต้องคิดวิเคราะห์ยังไง (Agent Engine) และค่อยสร้างอุอกมาเป็นคำตอบที่เชื่อถือได้

หัวใจของมันคือ 'ความน่าเชื่อถือ' ครับ เราอยากได้ระบบที่ตอบเหมือนเดิมทุกครั้งถ้าถามเหมือนเดิม ไม่ใช่ AI ารบณ์ศิลป์ที่วันนี้ตอบอย่างพรุ่งนี้ตอบอักษรย่าง และนี่คือสิ่งที่ UET Platform ถูกสร้างขึ้นมาเพื่อตอบโจทย์ครับ