FLORIDA ATLANTIC

COT 6930 - Generative Intelligence
and Software Development Lifecycle

Topic 5 -
Generative Intelligence Pipelines

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me

Agenda

Fundamentals of Generative AI Pipelines

Dynamic Orchestration

Adaptive AI Pipelines

Use Cases

Our Key Question

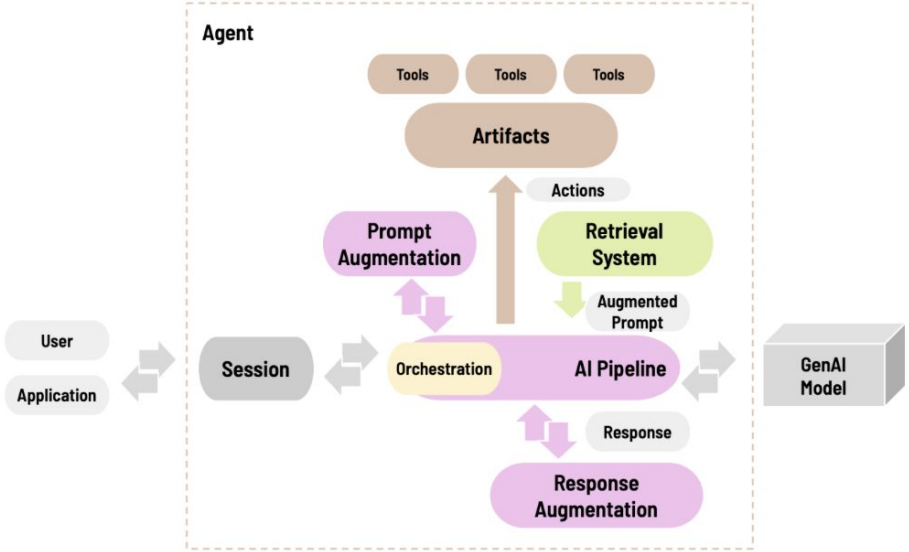
How can we design and manage AI pipelines that integrate multiple models, tools, and components to build intelligent, adaptive systems?

Fundamentals of Generative AI Pipelines

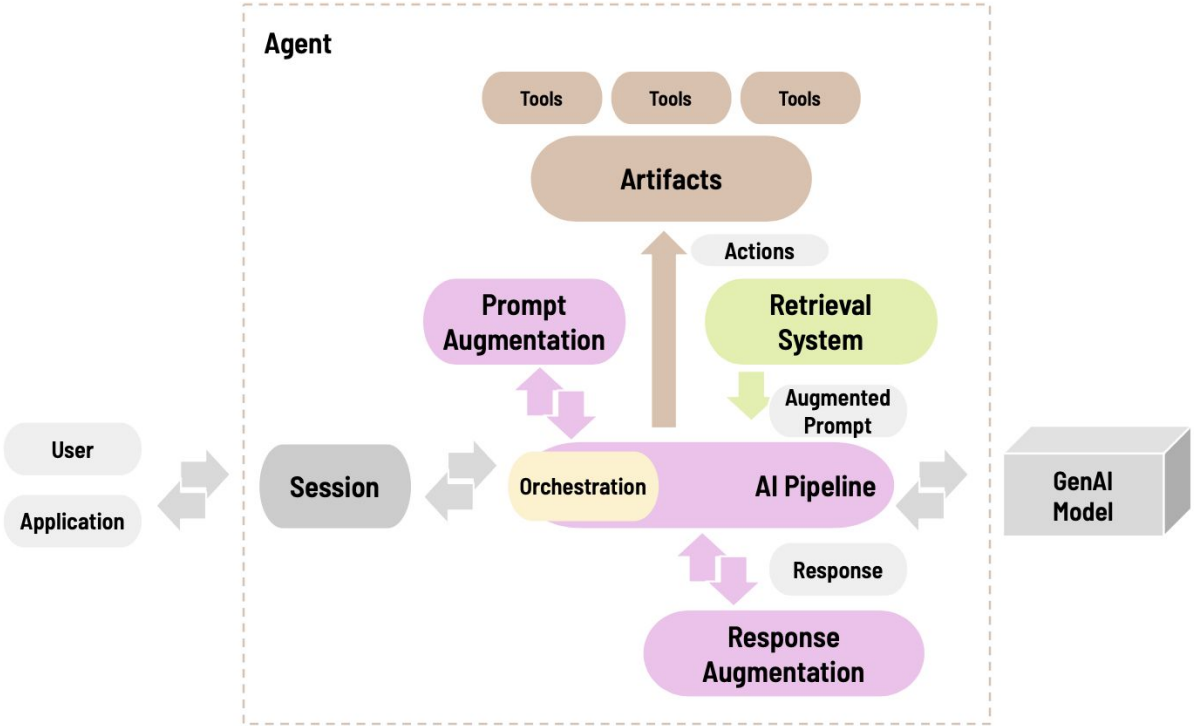
Generative Intelligence System

End-to-end systems with autonomous content generation

- Session-aware workflows.
- Automated prompt engineering.
- Context-aware content generation
- Adaptive behaviour combining orchestration, memory, retrieval augmentation, others.
- End-to-end automation for LGM consumption.



Systematic view of GenAI Pipelines

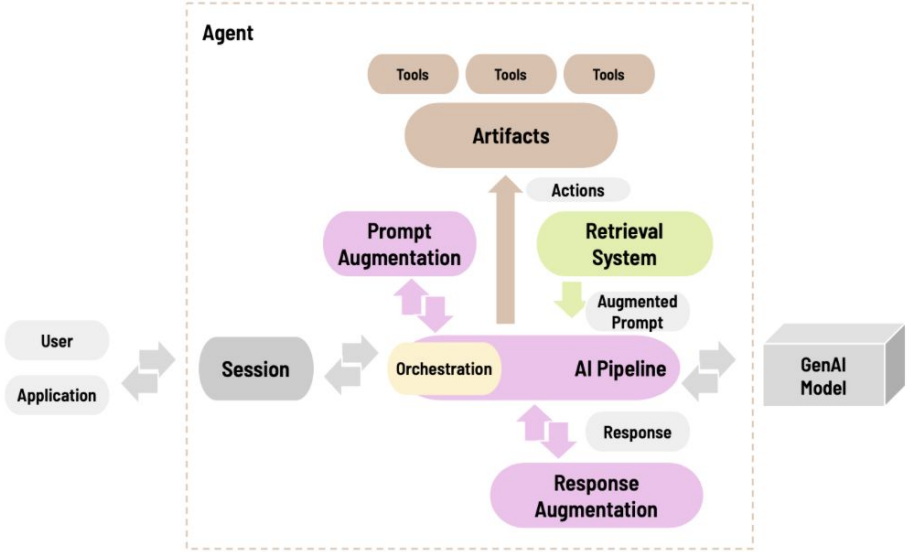


GenAI Pipeline

Structured workflow that connects AI models, tools, and knowledge sources to solve complex tasks.

- Orchestrate processing of prompt requests through Session → Prompt Augmentation → Retrieval Systems → Artifacts → Model Requests.
- **Answer to ‘What to do next?’**

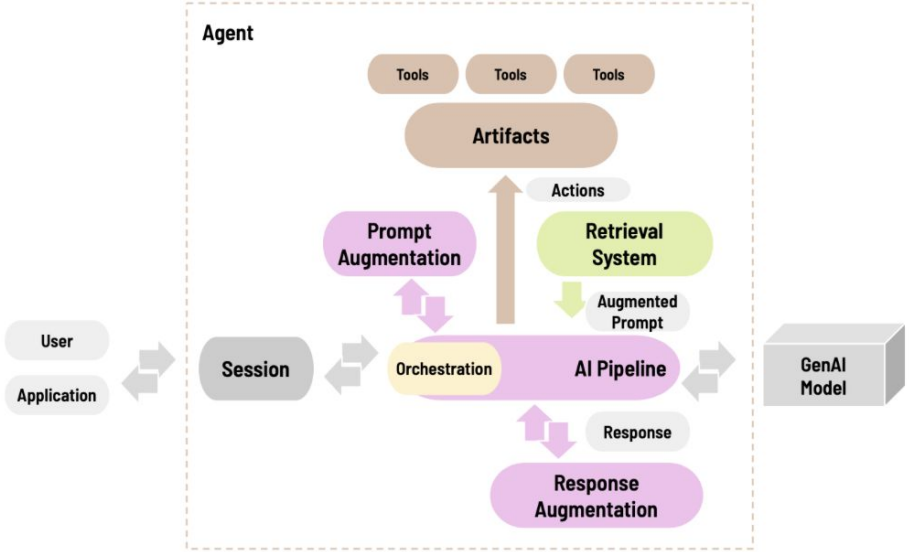
.



What are Sessions?

A session is the ongoing interaction between a user (or system) and the AI model.

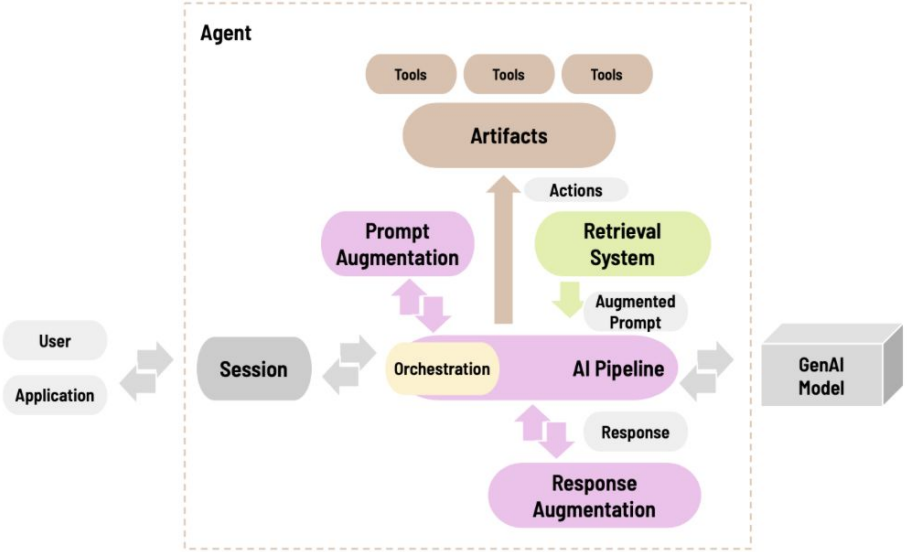
- Tracks the sequence of prompts and responses
- Maintains context across multiple turns
- Enables refinement and iteration over answers
- Critical for conversational AI and complex workflows



What are the Orchestration Mechanisms?

Coordinates how different pipeline components interact to produce coherent results.

- **Answer to ‘What to do next?’**
- **Flow Control:** Directs prompt requests through augmentation, retrieval, and response stages.
- **Adaptive Routing:** Dynamically chooses which systems (retrieval, tools, models) to invoke.
- **Error Handling & Validation:** Ensures robustness by refining or retrying when needed.



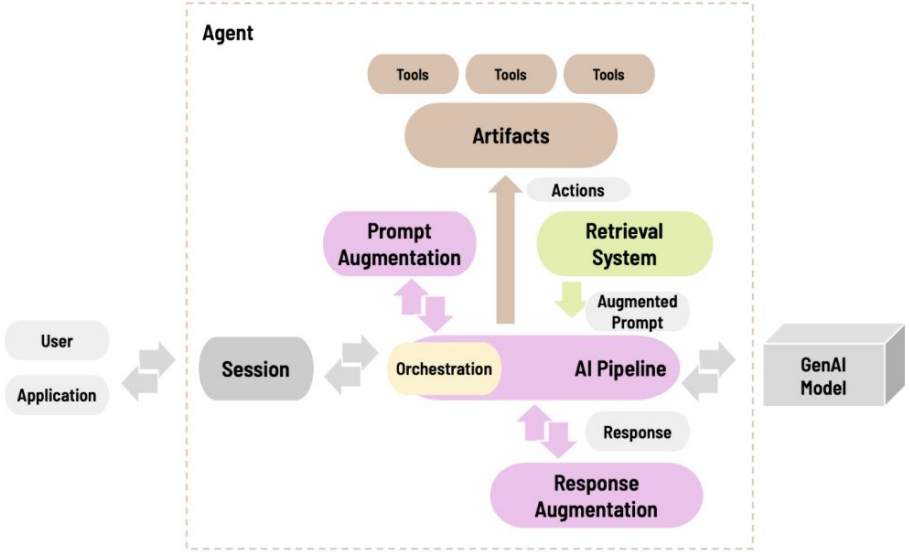
Static vs Dynamic Orchestration

Static Orchestration:

- Predefined, rule-based sequence of steps.
- Suitable for repetitive, predictable workflows.
- Example: *Always → Retrieval → Model → Response Augmentation.*

Dynamic Orchestration:

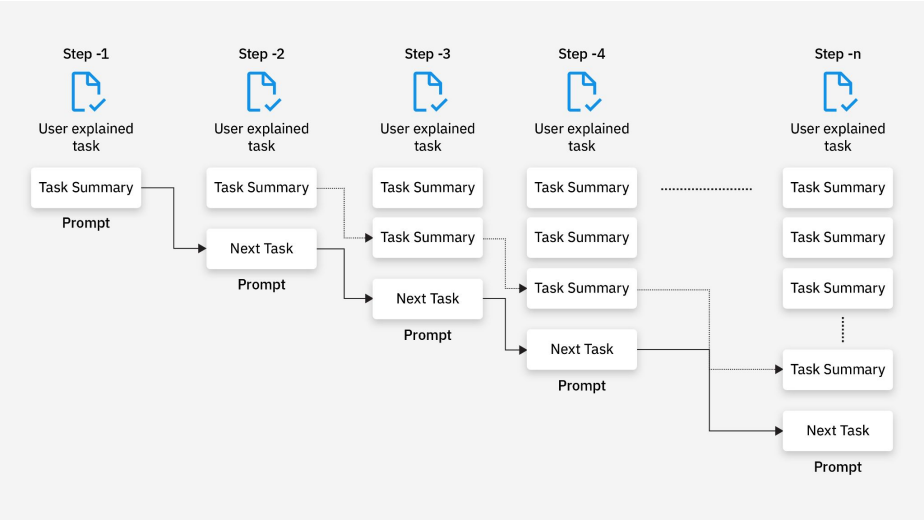
- Adaptive routing based on context, user input, or intermediate outputs.
- Enables flexibility and personalization.
- Example: *Skip retrieval if session memory already has needed info.*



What is Prompt Chaining?

Breaks complex tasks into smaller, sequential steps where each output informs the next.

- **Stepwise Reasoning:** Output of one prompt feeds into the next for structured progress.
- **Task Decomposition:** Simplifies large problems into manageable subtasks.
- **Context Building:** Each chain stage enriches the prompt with intermediate results.



LangChain and LangGraph

LangChain:

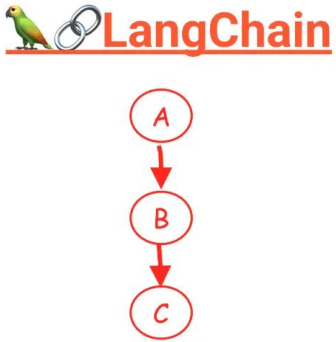
- Provides modular building blocks for chaining prompts, retrieval, tools, and models.
- Supports prompt templates, memory, and structured workflows.

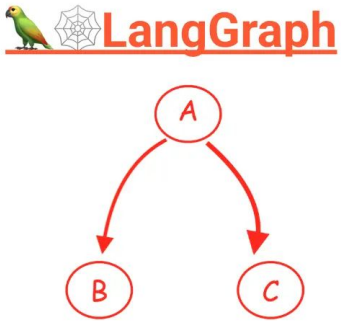
https://www.langchain.com/

LangGraph:

- Extends LangChain with graph-based orchestration of multi-step reasoning.
- Enables branching, looping, and conditional prompt flows.

https://www.langchain.com/langgraph



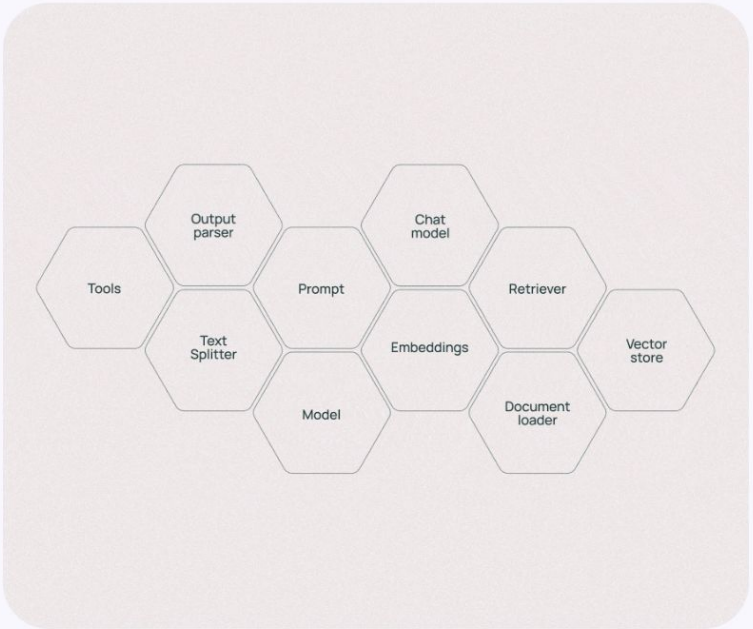


LangChain

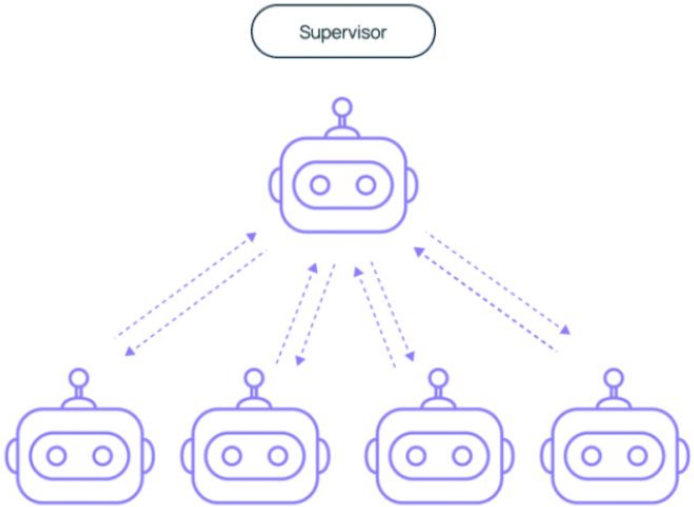
Why use LangChain?

Composable by design

LangChain's standard interface lets you experiment with different providers, tools, and databases – creating DevEx parity when gaps exist.



LangGraph



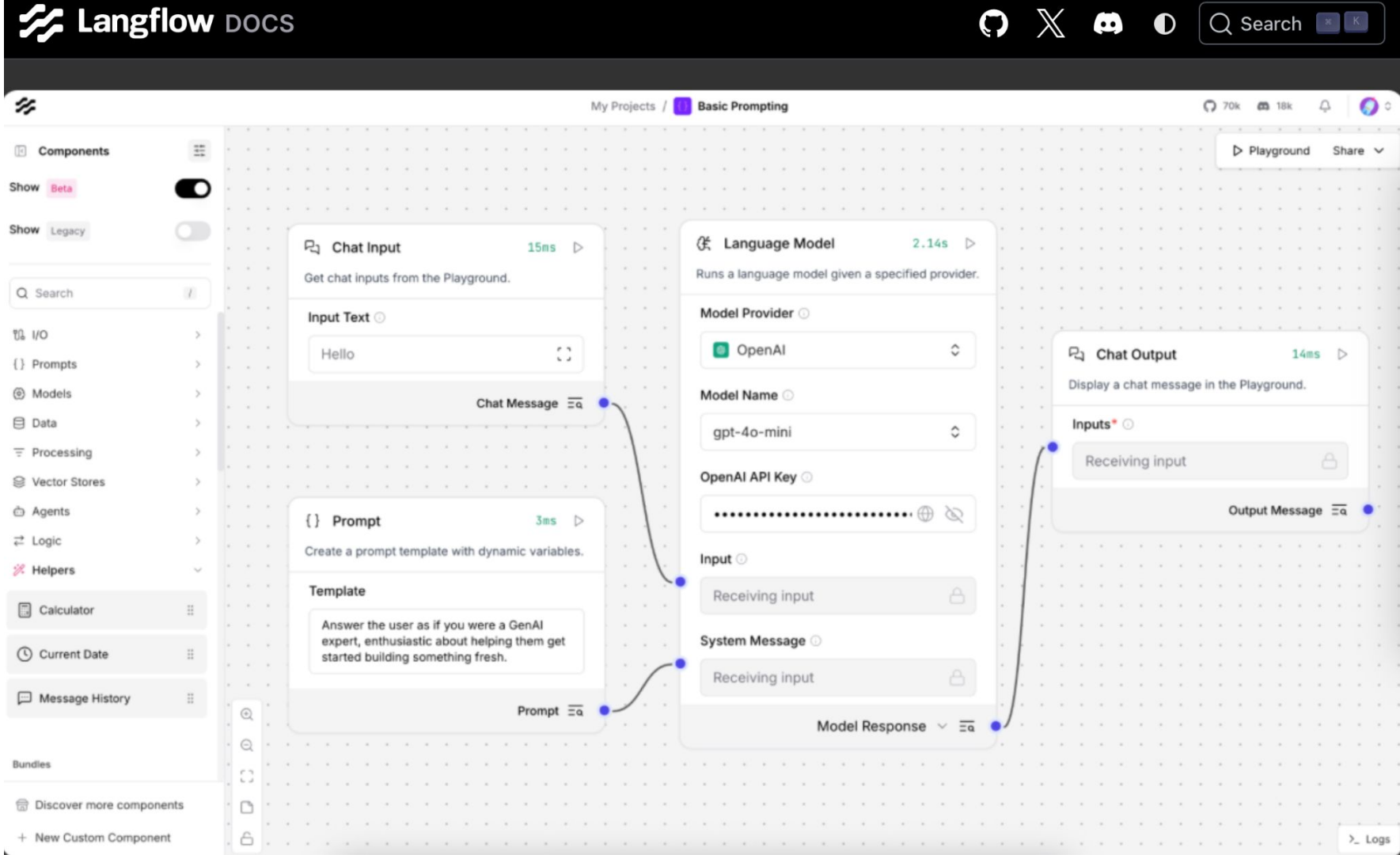
Build expressive, customizable agent workflows.

LangGraph's low-level primitives provide the flexibility needed to create fully customizable agents. Design diverse control flows — single, multi-agent, hierarchical — all using one framework.

See different agent architectures ↗

https://langchain-ai.github.io/langgraph/concepts/agentic_concepts/

LangFlow


Langflow DOCS
Search
My Projects / Basic Prompting
Playground
Share
Components
Show Beta
Show Legacy
Search
I/O
Prompts
Models
Data
Processing
Vector Stores
Agents
Logic
Helpers
Calculator
Current Date
Message History
Bundles
Discover more components
New Custom Component
Chat Input 15ms
Get chat inputs from the Playground.
Input Text
Hello
Chat Message
Prompt 3ms
Create a prompt template with dynamic variables.
Template
Answer the user as if you were a GenAI expert, enthusiastic about helping them get started building something fresh.
Prompt
Language Model 2.14s
Runs a language model given a specified provider.
Model Provider
OpenAI
Model Name
gpt-4o-mini
OpenAI API Key
Input
Receiving input
System Message
Receiving input
Model Response
Chat Output 14ms
Display a chat message in the Playground.
Inputs*
Receiving input
Output Message
Logs

LangFlow and Low-Code Solutions

Frameworks to manage AI pipeline execution.

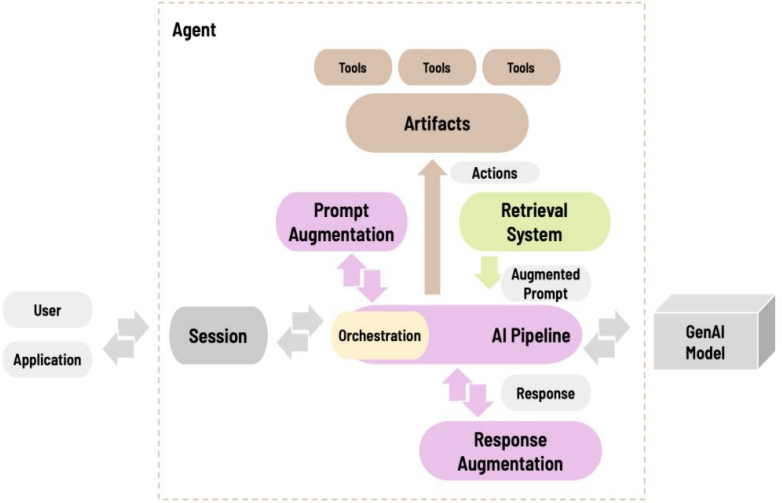
- Schedule, route, and monitor tasks
- Enable conditional branching and looping
- Support multi-step dependencies

- **LangChain**: modular chaining of model calls
- **LangGraph**: graph-based orchestration for adaptive flows

Dynamic Orchestration

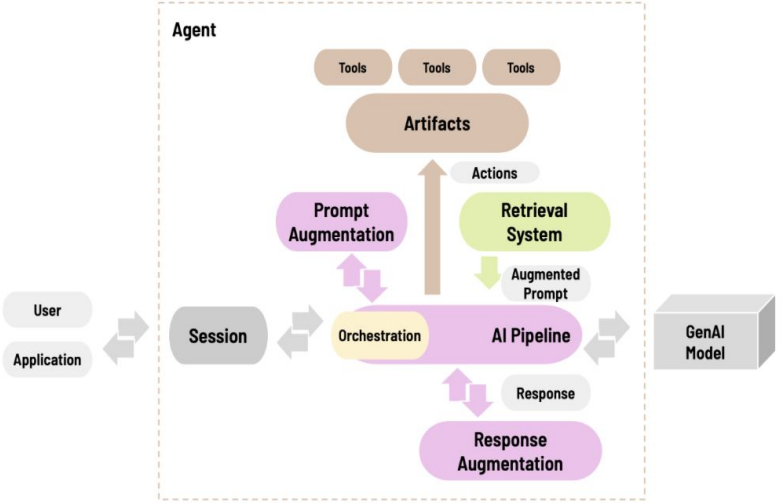
Where can we implement Dynamic Orchestration?

- **Session Context** – Adjust orchestration based on session state
- **Prompt Augmentation** – Use different strategies per task
- **Retrieval Systems** – Adaptive retrieval depth/scope
- **Model Parameters** – Dynamic tuning of generation settings
- **Model Selection** – Use specialized models for different requests
- **Response Augmentation** – Adapt presentation style



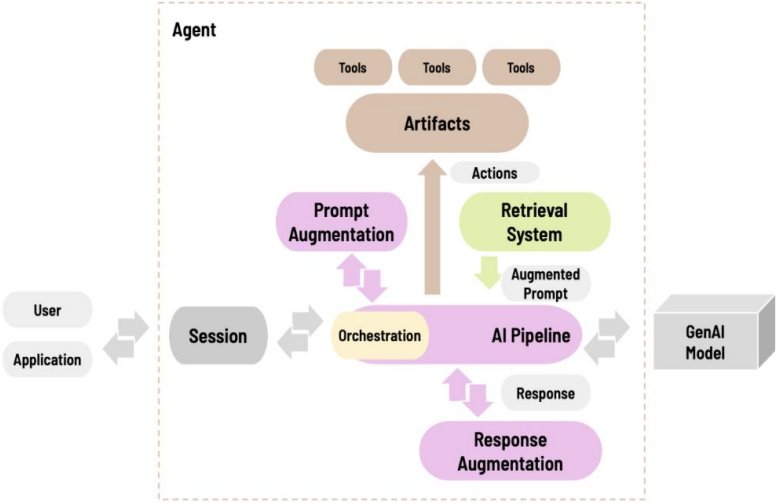
Dynamic Orchestration of Session Control (1/2)

- **Topic Continuation –** If the user continues on the same subject, then reuse prior responses as context.
- **Topic Reset –** If a new topic is detected, then clear memory or down-weight old session data.
- **Adaptive Memory Length –** If the conversation is long or short, then expand or shrink the context window accordingly.
- **Relevance Filtering –** If past turns are not relevant to the current query, then exclude them from the prompt.
- **Context Summarization –** If the session history is too large, then summarize it into compact memory before injecting.



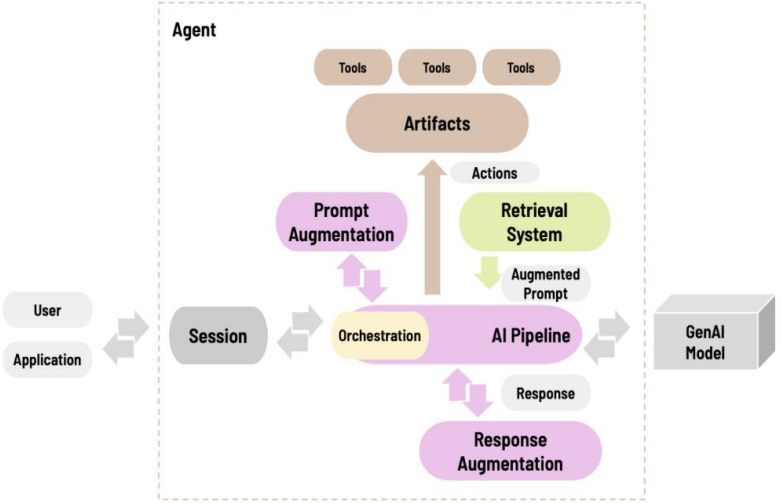
Dynamic Orchestration of Session Control (2/2)

- **Multi-Session Linking –** If a user revisits a previous topic in a new session, then recall relevant knowledge across sessions.
- **User Preference Adaptation –** If the session history shows tone, detail, or style preferences, then apply them to future responses.
- **Conflict Resolution –** If contradictions exist in prior session context, then reconcile or clarify before prompting.
- **Confidence-Based Use –** If confidence in the relevance of session memory is low, then exclude it from the prompt.



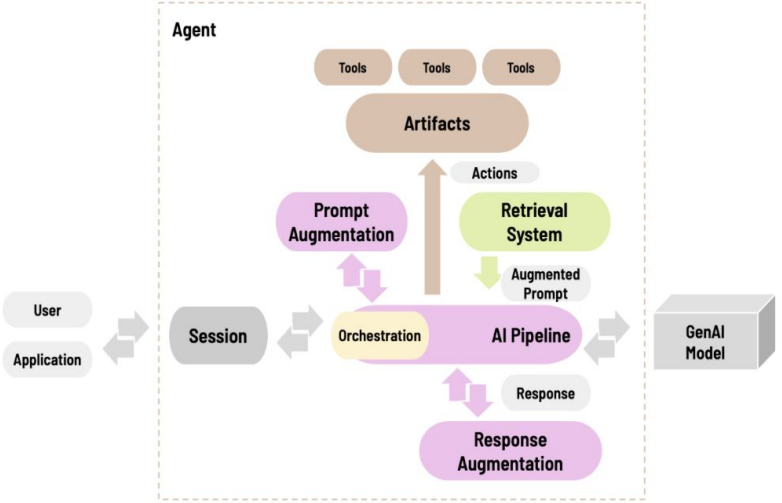
Dynamic Orchestration of Prompt Augmentation (1/2)

- **Template Switching –** If the task type is Q&A, summarization, reasoning, (other) then switch to the matching prompt template.
- **Role Conditioning –** If the task requires a specific persona, then add system role instructions (e.g., tutor, programmer).
- **Task Decomposition –** If the prompt is too complex, then break it into multiple chained sub-prompts.



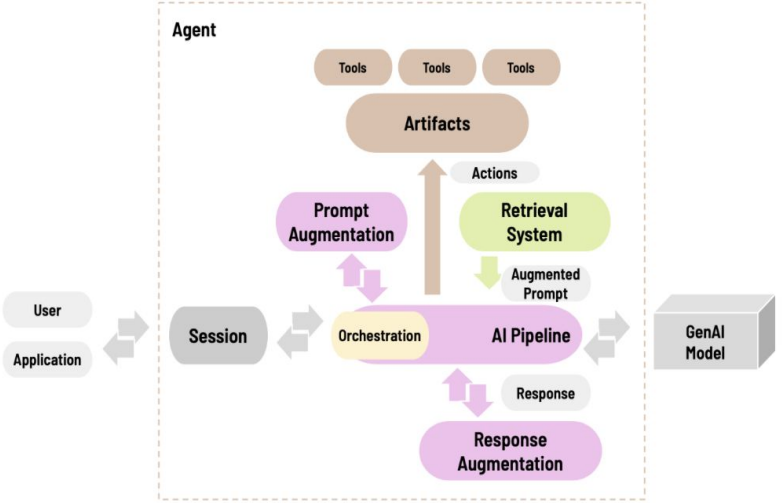
Dynamic Orchestration of Prompt Augmentation (2/2)

- **Style Adaptation –** If the session history shows a preference for tone, detail, or format, then adapt the style accordingly.
- **Multi-Perspective Framing –** If the task benefits from diverse reasoning, then reframe the prompt in multiple ways before calling the model.
- **Confidence Injection –** If the query is ambiguous or requires accuracy, then strengthen the prompt with validation or reasoning-check instructions.



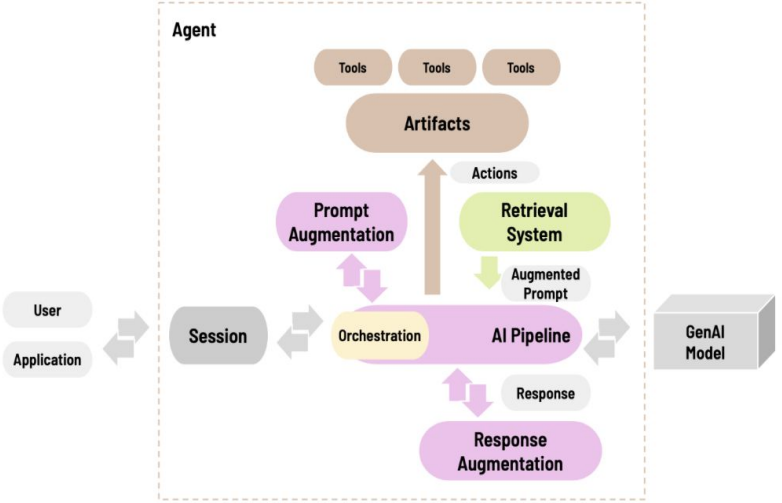
Dynamic Orchestration of Retrieval Systems (1/2)

- **Adaptive Retrieval Depth –** If the task is complex, then fetch more documents; if it is simple, then fetch fewer.
- **Source Switching –** If the query requires prior context, then use session memory; if it needs external facts, then use a knowledge base or APIs.
- **Hybrid Search –** If the query is factual, then prioritize keyword search; if it is conceptual, then prioritize semantic (vector) search.
- **Multi-Hop Retrieval –** If one retrieval step is insufficient, then perform chained retrieval across multiple sources.
-



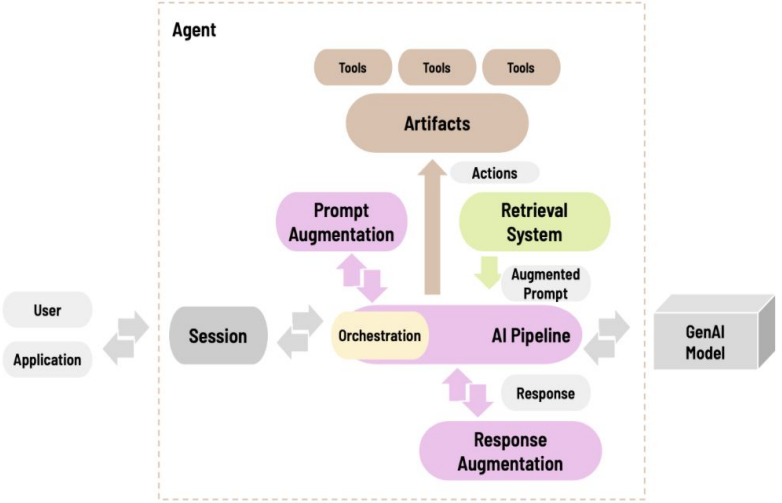
Dynamic Orchestration of Retrieval Systems (2/2)

- **Relevance Thresholding –** If precision is critical, then apply strict filters for highly relevant documents; if coverage is needed, then relax filters.
- **Domain-Aware Routing –** If the query is domain-specific (e.g., legal, medical, code), then direct it to the appropriate specialized source.
- **Freshness Prioritization –** If the task is time-sensitive, then prioritize newer sources over older ones.
- **Fallback Mechanism –** If the initial retrieval results are poor, then retry with alternative retrieval strategies.



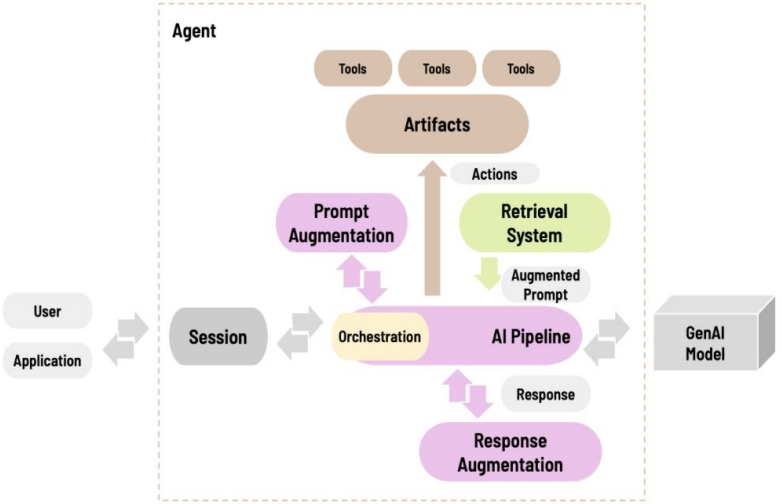
Dynamic Orchestration of Model Parameters (1/3)

- **Temperature Control –** If the task requires precision, then lower the temperature; if it requires creativity, then raise it.
- **Max Tokens Scaling –** If the task requires detailed output, then increase max tokens; if it requires brevity, then reduce them.
- **Decoding Strategy Switching –** If accuracy is needed, then use greedy decoding; if completeness is needed, then use beam search; if variety is needed, then use sampling.



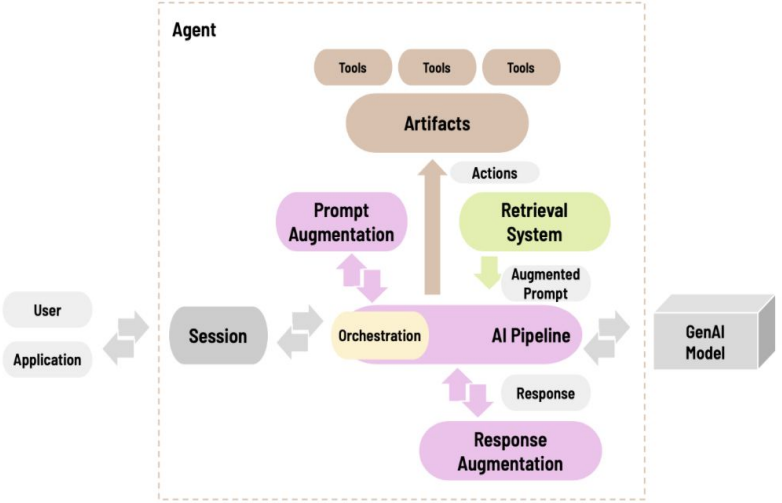
Dynamic Orchestration of Model Parameters (2/3)

- **Top-k / Top-p Tuning –** If the task requires controlled diversity, then adjust top-k/top-p values accordingly.
- **Detail Level Adaptation –** If the user prefers summaries, then limit verbosity; if they prefer depth, then expand explanations.
- **Confidence-Aware Parameters –** If confidence is low (e.g., weak retrieval scores or uncertainty signals), then increase reasoning depth (more tokens, add self-check/validation steps); if confidence is high, then keep default concise settings.



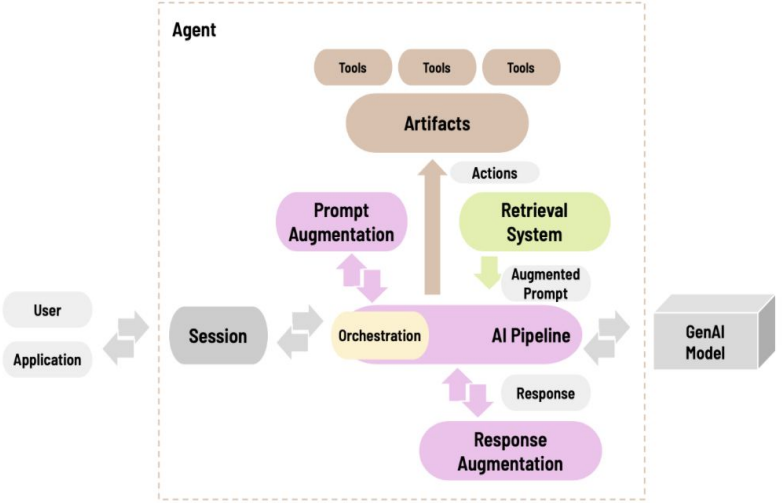
Dynamic Orchestration of Model Parameters (3/3)

- **Domain-Specific Settings –** If the task is technical/compliance-heavy, then lower temperature and enforce structured output; if it's creative, then raise temperature and allow freer sampling.
- **Progressive Refinement –** If generating from scratch, then draft with looser settings and refine with stricter ones; if polishing existing text, then start strict and iterate lightly.



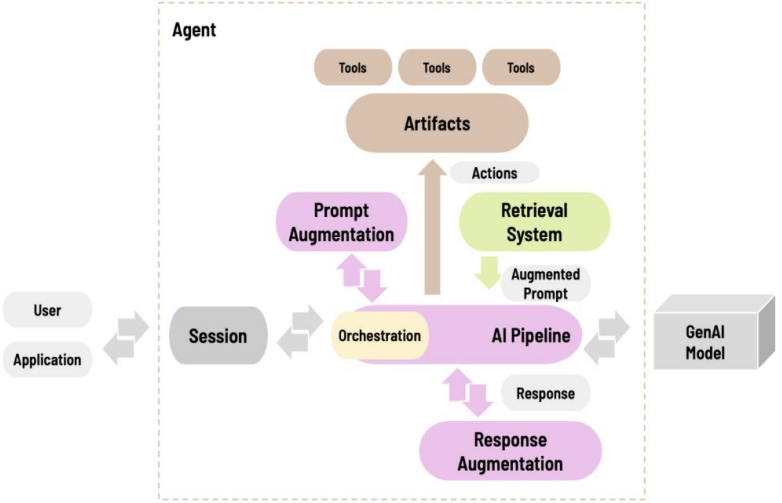
Dynamic Orchestration of Model Selection (1/2)

- **Lightweight Routing –** If the task is simple and factual, then use a smaller, faster model.
- **Complex Reasoning –** If the task involves multi-step reasoning, then route to a larger, more capable model.
- **Domain-Specific Choice –** If the query is in a specialized domain (e.g., legal, medical, coding), then select a domain-specific model.



Dynamic Orchestration of Model Selection (2/2)

- **Multi-Model Validation –** If accuracy is critical, then generate answers from multiple models and compare/aggregate.
- **Cost-Aware Selection –** If budget is constrained, then prioritize cheaper models unless high accuracy is required.



Adaptive Orchestration

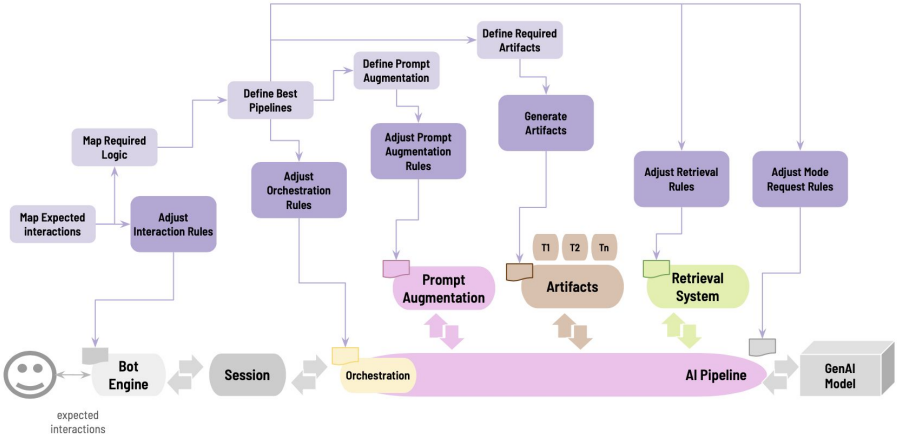
Can we automate the configuration of GenAI Pipelines?

Would automated configuration be a meaningful step towards advancing AI Solutions?

Why to Adapt AI Pipelines?

Adaptation makes AI pipelines more flexible, efficient, and personalized by dynamically adjusting to context and user needs.

- **Self-Programming:** pipelines automatically refine their own rules and strategies
- **Less Coding**: Reduces reliance on hard-coded logic
- **Flexibility:** Adjusts to new tasks and contexts on-the-fly
- **Efficiency:** Reduces wasted steps, latency, and resource costs.



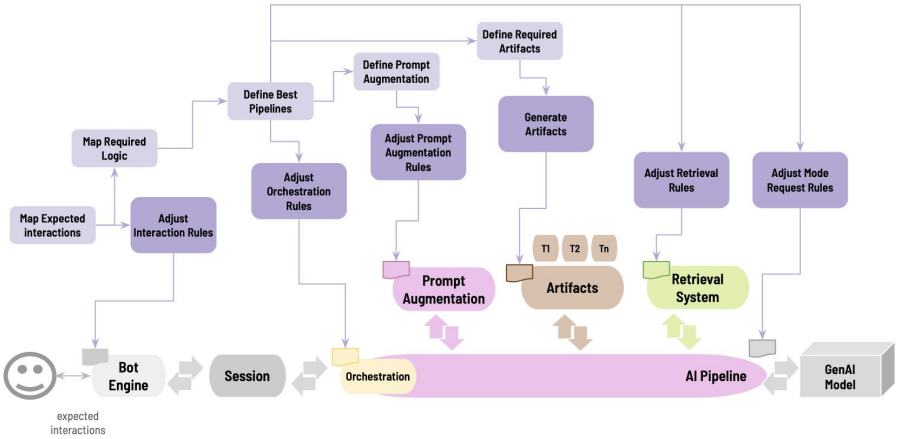
Why to Adapt in AI Pipelines?

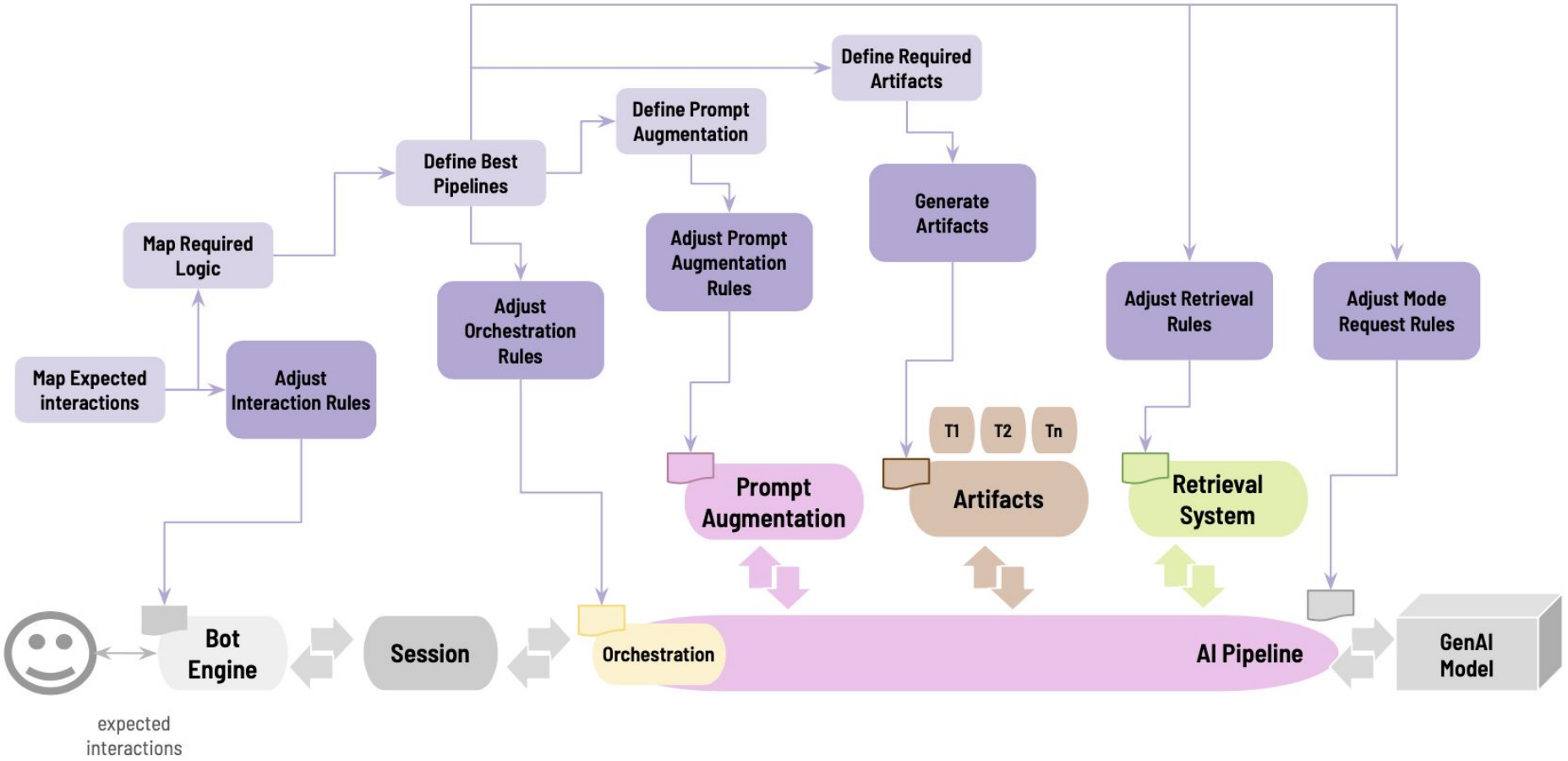
Adapt Prompt Augmentation Rules → How can prompts be automatically adapted to improve clarity, personalization, and reasoning?

Adapt Retrieval Rules → How can retrieval strategies adapt on-the-fly to provide the right knowledge at the right time?

Adapt Model Request Rules → How can the system create new model request strategies for emerging needs?

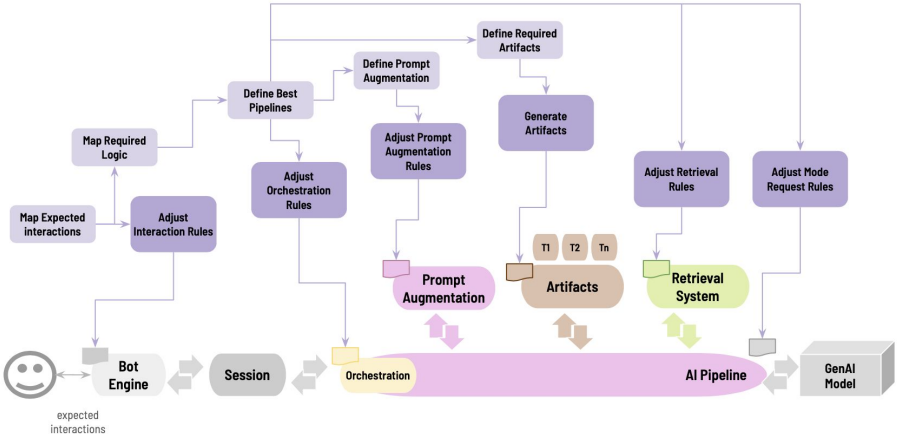
Generate Artifacts → How can the system generate new forms of artifacts to improve reasoning and continuity?




Define Required Artifacts
Define Prompt Augmentation
Define Best Pipelines
Map Required Logic
Adjust Prompt Augmentation Rules
Generate Artifacts
Adjust Retrieval Rules
Adjust Mode Request Rules
Adjust Orchestration Rules
Map Expected interactions
Adjust Interaction Rules
T1
T2
Tn
Prompt Augmentation
Artifacts
Retrieval System
Bot Engine
Session
Orchestration
AI Pipeline
GenAI Model
expected interactions

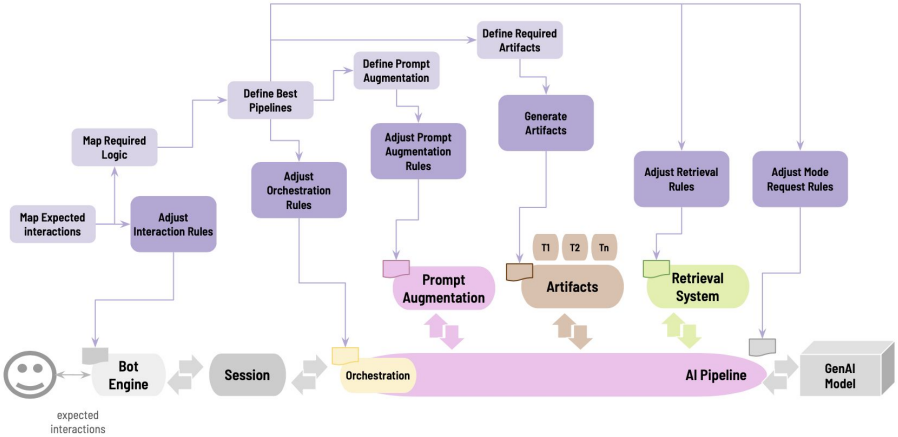
Adapt Interaction Rules (1/2)

- **Adjust Rule-Based Resolution** – If an interaction can be expressed as condition–action logic, then automatically generate the rules to answer it without AI Pipeline.
- **Optimization Enforcement** – If a rule-based path consistently resolves a query, then strengthen operational rules to bypass AI Pipeline.
- **Preference Adaptation** – If user history shows consistent style/format needs, then build rules that automatically enforce those preferences.



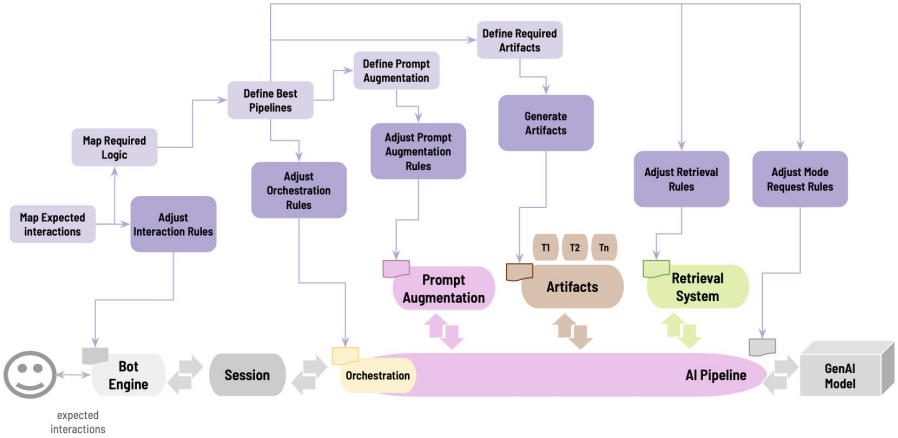
Adapt Interaction Rules (2/2)

- **Generate API/DB Mappings** – If a user request maps to structured data, then auto-create condition–action rules to call APIs or databases.
- **Optimize Workflow Triggers** – If a query aligns with a known process (e.g., ticket creation), then generate or update rules to directly trigger workflows.
- **Clarification Injection** – If ambiguity is detected in input, then produce a rule that inserts clarification prompts before escalating to AI.
- **Sentiment Conditioning** – If sentiment analysis flags frustration, then adjust or create rules for empathetic, pre-defined responses.



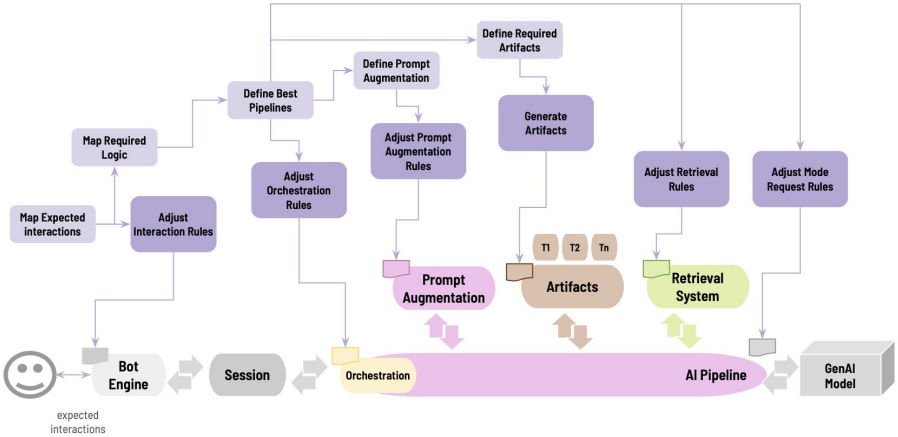
Adapt Orchestration Rules (1/2)

- **Path Prioritization –** If multiple orchestration paths can satisfy a task, then generate rules to prioritize the most efficient one.
- **Task Decomposition –** If a task requires multi-step reasoning, then create rules to chain prompts dynamically.
- **Intent Shift** – If user intent changes mid-task, then adapt rules to reroute execution.
- **Consensus Building –** If multiple outputs are available, then generate rules for validation and selection.



Adapt Orchestration Rules (2/2)

- **Load Balancing –** If workload spikes, then create rules to distribute across models/tools.
- **Latency Optimization –** If speed is critical, then create rules to skip optional steps.
- **Conflict Resolution –** If orchestration rules clash, then enforce precedence rules automatically.



EXERCISE

Exercise 5 - Getting Started with MyBot

Objective: Implement a modular AI pipeline, orchestrating data inputs, model inferences, and post-processing stages into a coherent flow.

Go to Canvas -> Assignments

COT 6930 - Generative Intelligence and Software Development Lifecycles

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me