260131 - Agentic AI Patterns Summary

Overview

- Introduction of Jigger Pathak as the primary trainer for the session focused on advanced agentic design patterns across multiple AML & AI trainee tracks.
- AI-first approach being adopted in insurance claims handling aims to cut resolution times and enhance customer satisfaction while automating complaint processes.
- Key challenges in current complaint systems identified as manual tracking, SLA violations, and lack of visibility into patterns, impacting efficiency.
- Proposed functional requirements include multi-channel intake with NLU for intent identification and an automated classification system with severity levels.
- Agentic AI architecture emphasizes distributed processing over traditional AI, enhancing performance through goal-oriented, multi-step reasoning.
- Introduction of reasoning frameworks like Chain of Thought and Plan and Execute, facilitating structured logical thinking and complex task management.
- Eight core Agentic Behavior Requirements (ABR) patterns highlighted, supporting goal-oriented planning and intelligent request routing based on intent.
- Proposed integrated use case showcases a comprehensive workflow for claims, including API for claim lookups and reflection processes for policy alignment.
- Multi-cloud architecture designs mapped out for Azure, AWS, and Google Cloud, utilizing respective services to enhance capabilities and security.
- Security considerations discussed encompass end-to-end security protocols, threat detection, and a fallback mechanism for conversation continuity in AI systems.

Notes

Session Introduction and Setup (21:32 - 28:02)

- Mixed audience session comprising AML track trainees, AWS AML trainees, Azure AML trainees, and Agent K and AFL trainees.
- Bonus session focus on advanced agentic design patterns, requiring trainees to review recordings afterward for comprehensive understanding.
- Jigger Pathak introduced as primary trainer for this architectural design session.
- Session clarification confirmed as common session for all tracks, replacing regular individual sessions for January 31st.

Business Objectives and Problem Statement (47:07 - 01:01:00)

- AI-first approach adoption by organizations for insurance claim complaint and resolution systems.

- Core business objectives include reducing claim resolution time, improving customer satisfaction, automating repetitive complaint handling, ensuring regulatory compliance, and supporting human-in-the-loop decision making.
- Current challenges identified as manual tracking, inconsistent responses, SLA violations, and limited visibility into complaint patterns.
- Stakeholder identification including policyholders, claim adjusters, customer support, compliance teams, IT/cloud teams, and data science/AI teams.

Functional Requirements and System Scope (01:01:06 - 01:22:15)

- Multi-channel complaint intake via chat, UI, or API gateway with free text complaint acceptance in local languages.
- Natural Language Understanding (NLU) for intent identification and entity extraction.
- Complaint classification system with severity levels and policy/claim data retrieval capabilities.
- Knowledge retrieval and decision recommendation engine with automated actions, human escalation for complex cases, and audit/compliance logging.

Agent AI Architecture and Patterns Overview (01:30:12 - 01:55:59)

- Traditional AI vs. Agentic AI comparison highlighting single model limitations versus goal-oriented multi-step reasoning with independent components.
- Agent composition defined as combination of LLM, planning, tools, memory, and user prompt creating autonomous agents.
- Microservices analogy explaining how Agent AI breaks down tasks into smaller pieces similar to application development patterns.
- Performance benefits achieved through distributed processing rather than overwhelming single models.

Reasoning Frameworks and Design Patterns (02:19:32 - 02:42:02)

- Chain of Thought pattern for step-by-step logical and mathematical reasoning before answering.
- React pattern alternating between reasoning and tool actions like a researcher who thinks, tests, observes, and concludes.
- Reflection pattern for self-evaluation and output refinement, particularly useful for code generation and writing improvement.
- Plan and Execute pattern breaking down complex multi-step tasks with manager-planner and engineer-executor collaboration.

Agentic Behavior Requirements (ABR) Framework (02:26:05 - 02:42:02)

- Eight core ABR patterns including planner-executor, router/dispatcher, RAG agent, policy guardrail, human-in-the-loop, reflection/self-check, tool-using agent, and stateful agent patterns.
- Goal-oriented planning decomposing complaint resolution into ordered steps: identify intent, retrieve claim details, check SLA, decide escalation, communicate outcome.

- Intelligent routing dynamically routing requests to specialized agents based on intent confidence with specialization meeting generalization principle.
- Knowledge grounded reasoning ensuring policy-related responses are grounded in retrieved documents with refuse-to-answer capability when no relevant documents found.

Integrated Use Case Implementation (02:34:41 - 02:42:02)

- Claim delay scenario workflow starting with user query processed through router, planner, tool execution, RAG retrieval, reflection, and final response.
- Tool integration capabilities including claim lookup API, CRM updates, and notification services for comprehensive system interaction.
- Reflection and verification process ensuring policy alignment, correct document citations, and accuracy before response generation.
- Architectural checklist requirements covering upstream execution, intent taxonomy, vector database for retrieval, rule engine for compliance, and governance frameworks.

Requirement to Architecture Mapping (02:42:09 - 02:46:08)

- Multi-channel intake mapping to Chat UI or API gateway with NLU agent for intent/entity extraction.
- Complaint classification handled by issue classification agent with RAG agent for policy/claim retrieval.
- Decision recommendations managed by decision and policy agent with action executor agent for automated actions.
- AI orchestration requirements including RAG vector store, rule engine decision making, escalation agents, and audit/observability layers.

Multi-Cloud Architecture Design (02:46:49 - 03:03:09)

- Azure implementation using Bot Service, API Management, AI Studio, AI Search, and Blob Storage.
- AWS architecture leveraging Amazon Lex for chatbot, API Gateway, Bedrock for models, OpenSearch for vector store, and S3 for document storage.
- Google Cloud setup utilizing Dialogflow CX, API Gateway, Vertex AI Agent Builder, Vertex AI Search, and Cloud Storage.
- API Management role providing security layer, rate limiting, policy enforcement, and request throttling between bot services and AI backend.

Security and Performance Considerations (03:05:51 - 03:14:06)

- End-to-end security approach including API management policies, private network communication, Microsoft Sentinel for threat detection, and service-level security standards.
- Fallback mechanism design for handling broken modules with memory integration for conversation continuity and alternative response paths.
- Cost optimization strategies comparing self-hosted LLM deployment vs. API-based solutions for processing millions of records with infrastructure considerations.

- Guardrails and compliance implementation using responsible AI systems, content safety services, and continuous response evaluation metrics.

Questions and Clarifications (03:03:09 - 03:29:19)

- Multi-agent vs. Agentic AI distinction clarified as multiple independent workers versus intelligent workers with brain, planning, and decision-making capabilities.
- Service mapping recommendations including Semantic Kernel Router, Azure Functions for routing, Azure AI Search for RAG, Cosmos DB for memory, and Azure AI Content Safety for guardrails.
- Batch processing considerations for insurance companies handling bulk claims with separate data processing workflows feeding into RAG systems for real-time query responses.
- Pattern standardization references to white papers and cloud provider documentation for established AI design pattern catalogs.

Summary

AI Design Patterns Training Session

The meeting served as an advanced bonus session for AIML track trainees, AWS, AzureML, and GCP trainees, focusing on AI design patterns and multi-cloud scenarios. Jigar, the trainer, clarified that this session was a one-time generic session to align participants on AI design thinking processes, and regular sessions would resume the following day. Participants raised questions about the session's timing and content, which were addressed by Karthik and Jigar, who explained the session's purpose and confirmed that the session was a one-time bonus session. Jigar also introduced a document on designing compound AI systems and emphasized the importance of understanding the design thinking process for AI solutions across various industries.

AI Insurance Claims System Design

Jigar led a session on designing AI-powered insurance claim complaint and resolution systems, focusing on multi-cloud implementation across Azure, AWS, and Google Cloud Platform. He explained the importance of understanding business objectives, problem statements, and current enterprise challenges when designing AI systems. Jigar outlined key components including reduced claim resolution time, improved customer satisfaction, automated repetitive complaint handling, regulatory compliance, and human-in-the-loop decision-making. He shared architectural blueprints and design patterns for implementing AI solutions, emphasizing the need for a structured approach to system design and integration of existing resources like policies and documents.

AI-Driven Insurance Claims System

Jigar discussed the design and implementation of an AI system to address high-volume complaints and issues related to insurance claims. He outlined key components such as automated intent detection, policy and claim data retrieval, decision recommendations, and human escalation for complex cases. Jigar emphasized the importance of understanding the scope of the AI system and the need for audit and compliance logging. He also highlighted the need to consider various stakeholders involved in the process and the potential for fraud detection and payment gateway integration in future iterations.

Insurance Claims Automation Design Framework

Jigar discussed the importance of understanding stakeholders and functional requirements when designing a solution for insurance claim and support automation. He outlined key stakeholders including policyholders, claim adjusters, customer support, compliance teams, IT and cloud teams, and data science/AI teams. Jigar explained several functional requirements including complaint intake channels, natural language understanding, complaint classification, and policy/claim data retrieval. He emphasized that the focus of this session is on architecture and design thinking, not direct implementation, as companies typically first understand the system before implementing it.

Agentic AI Design Thinking Overview

Jigar discussed the design thinking process and its importance in understanding and implementing AI solutions, emphasizing the need to address both theory and implementation aspects. He explained the concept of agentic AI and its advantages over traditional AI, particularly in handling complex tasks and improving system performance. Jigar also touched on the use of AI in cybersecurity and the potential for implementing AI solutions in various industries beyond insurance and claims processing. The session concluded with a plan to take a break and continue the discussion on reasoning frameworks and architectural components for AI solutions.

GenAI vs Traditional App Development

Jigar discussed the decision-making process between using GenAI agents versus traditional app development, focusing on cost considerations and the need for self-directed, automated, goal-driven activities. He shared white papers and links to help participants understand the rise of GenAI and its applications, emphasizing the importance of reasoning frameworks in AI development. Jigar also explained various reasoning patterns, such as chain of thought, react and reason plus action, and the components of an AI agent (LLM, planning, memory, tool, and user prompt). He concluded by outlining agentic behavior requirements and patterns, such as planner agents, router-dispatcher agents, and the importance of goal-oriented planning in AI systems.

AI Agent Implementation Patterns

Jigar presented a structured approach to implementing AI agents, breaking down complex patterns into separate components like planner, router, RAG, and guardrails. He explained how these patterns can be applied to specific use cases, such as handling claim delays, by integrating various tools and maintaining stateful multi-turn reasoning. Jigar emphasized the importance of design thinking and visualizing each pattern separately to avoid confusion, sharing a document with graphical representations to aid understanding.

AI-Driven Complaint Intake Architecture

Jigar presented an architectural overview of a multi-channel complaint intake system, detailing how user inputs would be processed through various AI agents including natural language understanding, issue classification, and knowledge retrieval components. He explained the technical implementation across different cloud providers - Azure, AWS, and Google - highlighting how each platform's specific services like Azure AI Studio, Amazon Lex, and Dialogflow CX would be utilized to create the end-to-end solution. Jigar clarified that Microsoft AI Studio (formerly AI Foundry) provides a unified development experience for generative AI applications, distinguishing it from AWS SageMaker which is more focused on machine learning.

AI Architecture for Insurance Claims

The meeting focused on discussing AI architecture patterns, particularly in the context of insurance claim processing. Jigar explained various components including routers, planners, and reflection agents, and how they can be implemented using Azure AI services. He also covered topics like API management for request throttling, security considerations for the pipeline, and the use of LLMs for processing large volumes of data. The discussion included questions about cost-effectiveness of different deployment options and the mapping of AI patterns to specific Azure services. Jigar emphasized the importance of understanding the difference between multi-agent systems and agentic AI, and promised to share additional resources and documentation for further learning.