Design Patterns for Agentic AI:

Building Scalable, Event-Driven Systems

InfoQ Webinar Q&A Transcript

Q: **How can the agents be integrated into a complex legacy enterprise architecture? Are there any best practices that you can suggest?**

A: To integrate agents into complex legacy environments, start with domain-driven design to identify where agents can best interoperate. Common approaches include:

- Working with important data that needs to be summarized, synthesized, analyzed, or reviewed.
- Feeding events (such as traces, logs, or workflow signals) into agentic systems to enable ambient AI models that make recommendations or drive decisions.
- Replacing complex rule-based systems with reasoning-based systems powered by LLMs (Large Language Models).

Q: **Is Akka providing a framework specifically for creating agentic multi-agent systems, or will it be Akka 3 as it is today augmented by the developer with external libraries to be able to work with LLMs and agents?**

A: Both. Currently, Akka SDK supports multi-agent orchestration through endpoints and workflows and includes native support for MCP and A2A protocols. However, we are evolving it further with "agentic abstractions" to reduce overhead and make development simpler.

Additionally, we are developing an open-source prompt language for creating agents. This will integrate with IDEs like Kudu, Winsurf, or Cursor to generate distributed systems directly from prompts.

Q: **Is it similar to n8n (n8n.io), which offers similar functionality as a workflow automation platform that uniquely combines AI capabilities with business process automation?**

A: Akka's agentic AI platform integrates durable workflow automation functionality that is similar to n8n. The platform also integrates memory, streaming, and resilience, all of which are critical for modern agentic applications. These capabilities are not part of n8n.

Q: **What is the difference between A2A and MCP?**

A: MCP (Model Context Protocol) allows agents to access tools and extend their capabilities (e.g., operating a Salesforce API or retrieving database content). A2A (Agent-to-Agent), proposed by Google, defines how agents discover, communicate, and cooperate with each other. They are complementary: MCP focuses on agent-to-system interaction, while A2A focuses on agent-to-agent interaction.

Q: **How does Akka compare with other tools/platforms enterprises may already have such as Dataiku, Databricks etc. who are also building such capabilities?**

A: Akka is an agentic developer platform, used by developers to code and deploy agentic AI systems. Both Databricks and Dataiku are data management / data warehouse solutions. This is generally complementary to Akka. Typically, you would use Databricks / Dataiku to curate and analyze your data before using it to fine-tune your AI model. You would then use Akka to build an agentic application that took advantage of your AI model.

Q: **For agentic workflows where tool calling chain or "next-step" is not pre-known, are there any auth/auth frameworks to think about for these agents that fall in line with accepted security measures and enterprise identity providers?**

A: Not yet. Today, many emit workflows as code that includes authentication hooks, allowing integration with enterprise authentication flows. Enterprise authentication remains a key gap in MCP, though this is expected to be addressed before the end of the year.

Q: **How do you take a problem-first approach to agentic systems?**

A: Agentic systems are orchestrated agents, which are a distributed systems problem. Agents rely upon models which are stochastic, have variance, and randomness. We have two design principles that drive how Akka systems behave and the experience you should expect:

1. Akka services embrace failure, leveraging location transparency, self-healing, and clustering-from-within to be able to recover from any network or hardware failure. This is enabled with Akka's runtime that automatically provides resilience and replication to all Akka services.

2. Akka AI agents embrace randomness, leveraging evaluation-driven design to provide validation, measurement, and deviation of accuracy, enabling human systems to monitor and optimize the accuracy of a system to build layers of certainty over time.

Q: **Is there a sandbox playground to play with Akka agentic SDK?**

A: Yes. The SDK can be used entirely offline with only Java and Maven required. You can also create a free account at Akka.io to deploy and scale agentic samples, which are packaged as Docker solutions. However, environments like GitPod or Coder.com are not yet enabled.

Q: **Will we be writing Actors in Akka agentic AI services?**
Akka's AI SDK enables you to build agents, endpoints, workflows, timers, consumers, views, and memory stores using procedural constructs that are then executed as actors. No knowledge or experience with actors is required to write these components, but your service will contain all their amazing goodness under the cover.

Q: **Will the LLM run in distributed mode to take advantage of Akka? If so, how will it handle collecting and sending context (such as calling tools) and direct that context to the LLM in distributed mode?**

Akka's integrated multi-region capabilities allow seamless integration with distributed LLMs. All memory and context is automatically replicated across regions. Akka will assemble appropriate context from locally replicated memory, and route that request to the nearest LLM.

Q: **How do we evaluate agent and agentic systems?**

In general, a multi-faceted approach is required for evaluation.

1. Curate a wide range of inputs. This includes logging real-world inputs, creating synthetic inputs (possibly with the help of a high quality LLM), and adversarial inputs.
2. Constantly measure the effectiveness of the agentic system against these inputs. This includes human feedback, using an LLM-as-judge, and, if possible, creating a clear scoring mechanism.

Q: **What are the recommendations for an agentic system to implement security by design and compliance?**

Yes - see slide 20.