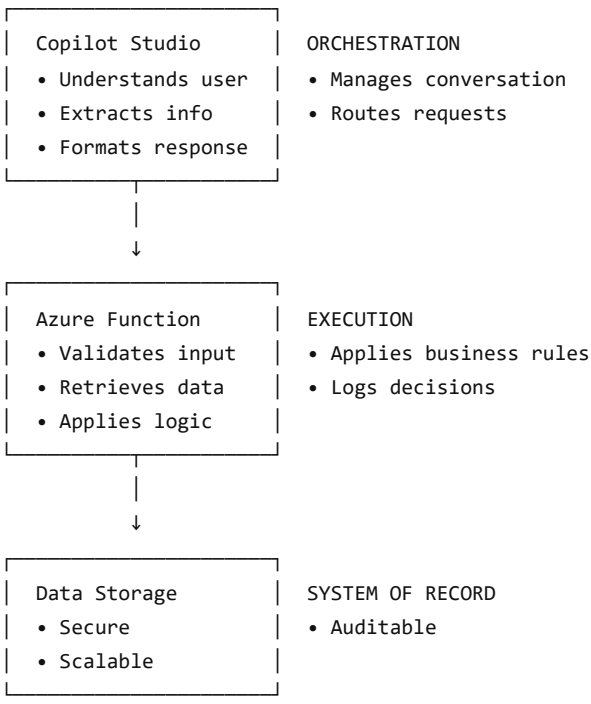
Building Agentic Systems with Copilot Studio

Session 2 - Audience Handout

The Core Principle

> ***AI is excellent at understanding intent and generating natural language. Code is excellent at making decisions. Use each tool for what it's best at.***



Why This Architecture?

Embedding Logic in Prompts	Using Azure Functions
Non-deterministic (varies each time)	Deterministic (same input = same output)
Prone to hallucination	Returns only real data
No audit trail	Full logging & traceability
Hits token limits at scale	Scales to millions of records
Hard to test	Unit testable
Prompt injection vulnerable	Secure by design

Demo Test URLs

Test the Azure Function directly:

```
https://func-orderstatus-demo.azurewebsites.net/api/GetOrderStatus?code=-
Rk7cIVaiBBG2OhnYxqdOdZwN4D8fPFSyC7NxApfvTu0AzFuIj9vfw==&orderId=ORD-1003
```

Test Orders:

Order ID	Expected Result
ORD-1003	Delayed order with 10% discount
ORD-9999	"Order not found" error
1001	"Invalid format" error

Copilot Studio Building Blocks

Component	Purpose
Topics	Entry points and flow control
Entities	Extract structured data from conversation
Actions	Call Power Automate, Azure Functions, APIs
Memory	Short-term context (not for data storage)
Triggers	Events from Dynamics, Dataverse, email, queues

When to Use Azure Functions

Use Azure Functions when:

- Business logic is complex or conditional
- Secure access to systems is required
- Data validation is needed
- Auditability and compliance matter
- Workflows are long-running or async

Rule of thumb: If a decision affects money, compliance, or data integrity → use code, not prompts.

Common Patterns

Multi-Agent Handoffs

Copilot Studio coordinates specialized downstream agents (Orders, Returns, Support) - each handles its domain.

Async Processing

For long-running tasks: send to Azure Queue → process in background → notify when complete.

Durable Functions

Multi-step workflows with checkpoints that survive failures. Retry from last checkpoint, not the beginning.

Escalation Paths

Never fully automate decisions requiring human judgment. Create approval requests and pause until approved.

Production Readiness Checklist

Security

- [ ] Use managed identities (not connection strings)
- [ ] Store secrets in Azure Key Vault
- [ ] Validate all inputs in functions
- [ ] Redact PII before sending to LLM

Cost Management

- [ ] Reference data via APIs (don't copy into prompts)
- [ ] Remove redundant context
- [ ] Monitor token usage with alerts

Observability

- [ ] Log inputs/outputs (with redaction)
- [ ] Monitor latency and error rates
- [ ] Use correlation IDs for tracing

Resilience

- [ ] Implement retries with backoff
- [ ] Set appropriate timeouts
- [ ] Design graceful degradation

Key Takeaways

1. **Copilot Studio = Orchestrator**, not a rules engine
2. **Prompts suggest**, code decides
3. **Deterministic behavior** requires code, not AI
4. **Audit trails** come from functions, not conversations
5. **Scale** requires data APIs, not hardcoded prompts
6. **Security** means validation in code, not instructions in prompts

---

Common Questions

Q: Why not just use AI for everything? A: AI excels at understanding language and conversation. Code excels at precise decisions. Use each for its strengths.

Q: Isn't this more complex? A: Initially, yes. But it's more maintainable, auditable, secure, and reliable. Complexity in the right place beats fragility everywhere.

Q: What about latency? A: Azure Functions in the same region add <100ms. Worth it for deterministic behavior and auditability.

Q: Can agents call other agents? A: Yes - multi-agent handoffs. Copilot coordinates, specialized agents execute.

Resources

- **Copilot Studio**: copilotstudio.microsoft.com
- **Azure Functions**: portal.azure.com
- **Power Automate**: make.powerautomate.com

Contact

Questions about implementing these patterns? Reach out to your session presenter or the Park Place Technologies team.

Session 2: Building Agentic Systems - Park Place Technologies