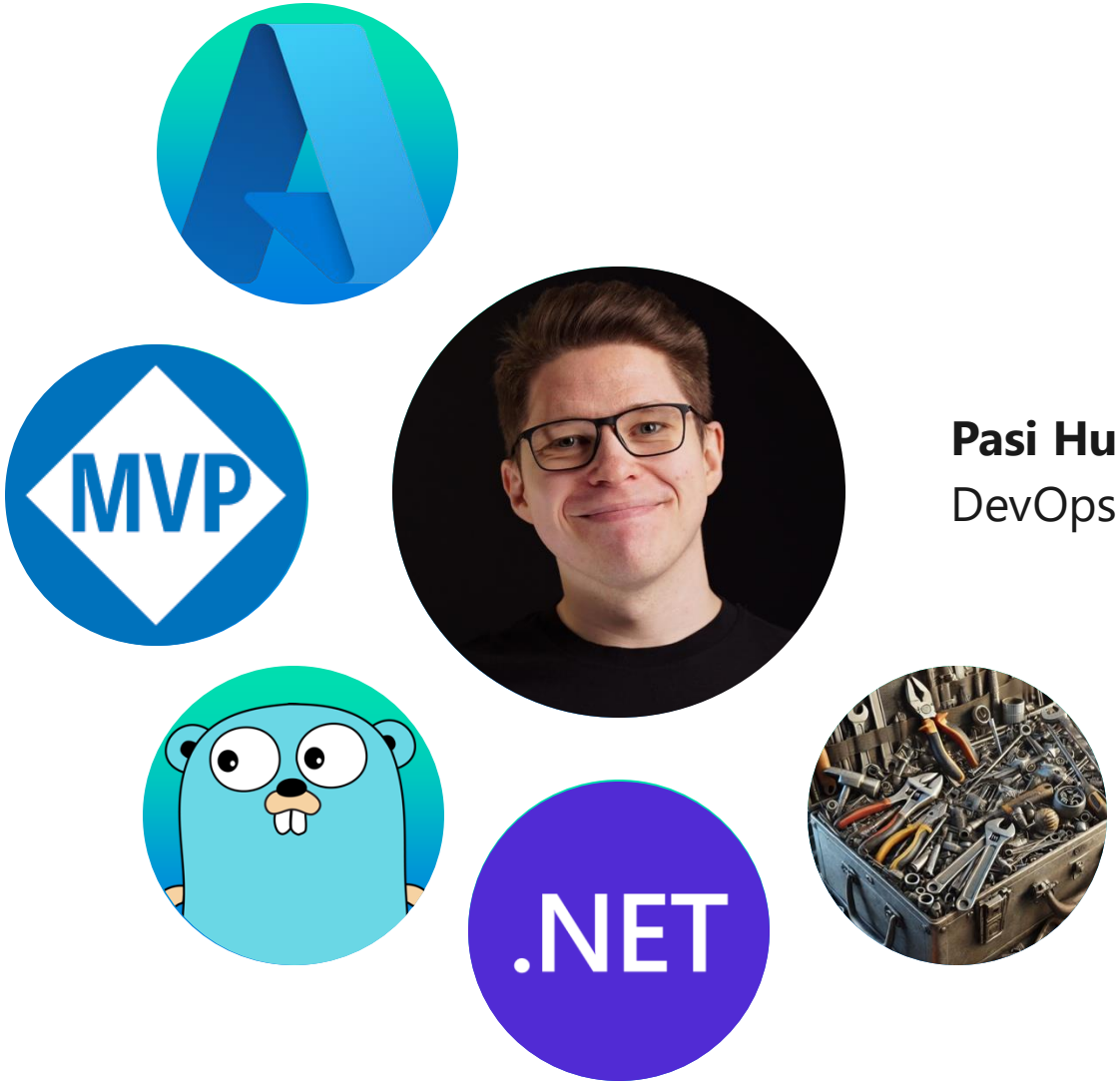
Multi-agent Orchestration Patterns in Semantic Kernel

ZURE



Pasi Huuhka
DevOps Architect @ Zure

ZURE

ZURE

- We solve business problems with technology
- Designing, building and managing on Azure since 2011
- 120+ experts with an average of 15 years of experience
- Offices in Finland, Belgium, Denmark, UK & Netherlands
- Fully independent, owned by employees

Microsoft Solutions Partner
Digital & App Innovation
Azure

Specialist
AI Platform on Microsoft Azure
Build AI Apps on Microsoft Azure
Migrate Enterprise Applications to Microsoft Azure

Microsoft Solutions Partner
Data & AI
Azure

Specialist
Analytics
Build AI Apps on Microsoft Azure
AI Platform on Microsoft Azure
Migrate Enterprise Applications to Microsoft Azure

Microsoft Solutions Partner
Infrastructure
Azure

Specialist
Azure Virtual Desktop

Microsoft Solutions Partner
Security

Specialist
Cloud Security
Threat Protection

MVP Microsoft® Most Valuable Professional

Microsoft Partner
Microsoft
2023 Partner of the Year Winner
Application Innovation and Modernization
Finland

In this presentation...

- Very short intro to Semantic Kernel
- Overview of SK orchestration patterns: Sequential, Concurrent, Group Chat, Handoff, Magentic
- What each pattern enables and when to use it in real workflows
- How to implement these patterns in C#

ZURE

Semantic Kernel

- Microsoft's **production ready** SDK for building Pro-Code AI applications
- Works with **many AI providers and copilots** (Copilot Studio, AI Foundry, AWS Bedrock, OpenAI, etc.).
- Lets you create **extensible agents** with plugins, filters, and controlled execution
- Designed for **developers**: available in .NET, Python, Java
- Built-in support for memory, embeddings, observability, prompt templates and much more
- **More control**, **more features** vs tools like Azure AI Foundry Agent Service or Copilot Studio

Agents

```
var scaffolder = AgentUtils.Create(
    name: "Scaffolder",
    description: "Suggests initial project structure, layers, and TODOs to get started.",
    instructions: "Describe the service skeleton and TODOs.",
    kernel: kernel);
```

ZURE

Kernel & Plugins

```
kernelBuilder.AddAzureOpenAIChatCompletion(
    deploymentName: options.Deployments.Llm,
    endpoint: options.Endpoint,
    credentials: credential);
```

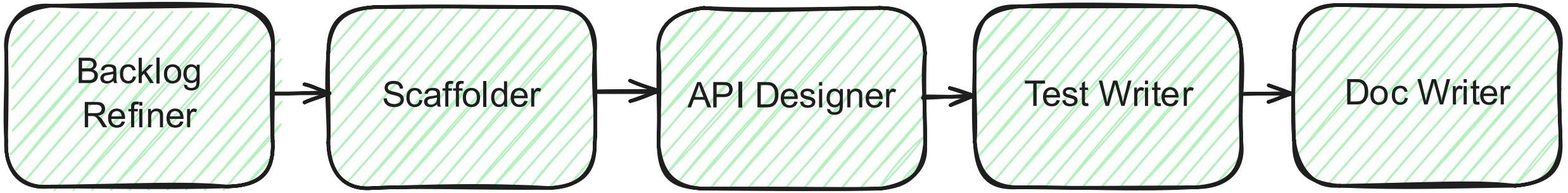
```
public sealed class DevWorkflowPlugin
{
    [KernelFunction, Description("Generate OpenAPI from story and AC")]
    0 references
    public string Oas_Generate(
        [Description("story id")] string story,
        [Description("acceptance JSON")] string acceptance)
        => "DO SOMETHING HERE";
```

```
designAgent.Kernel.ImportPluginFromObject(new DesignPlugin(), nameof(DesignPlugin));
```

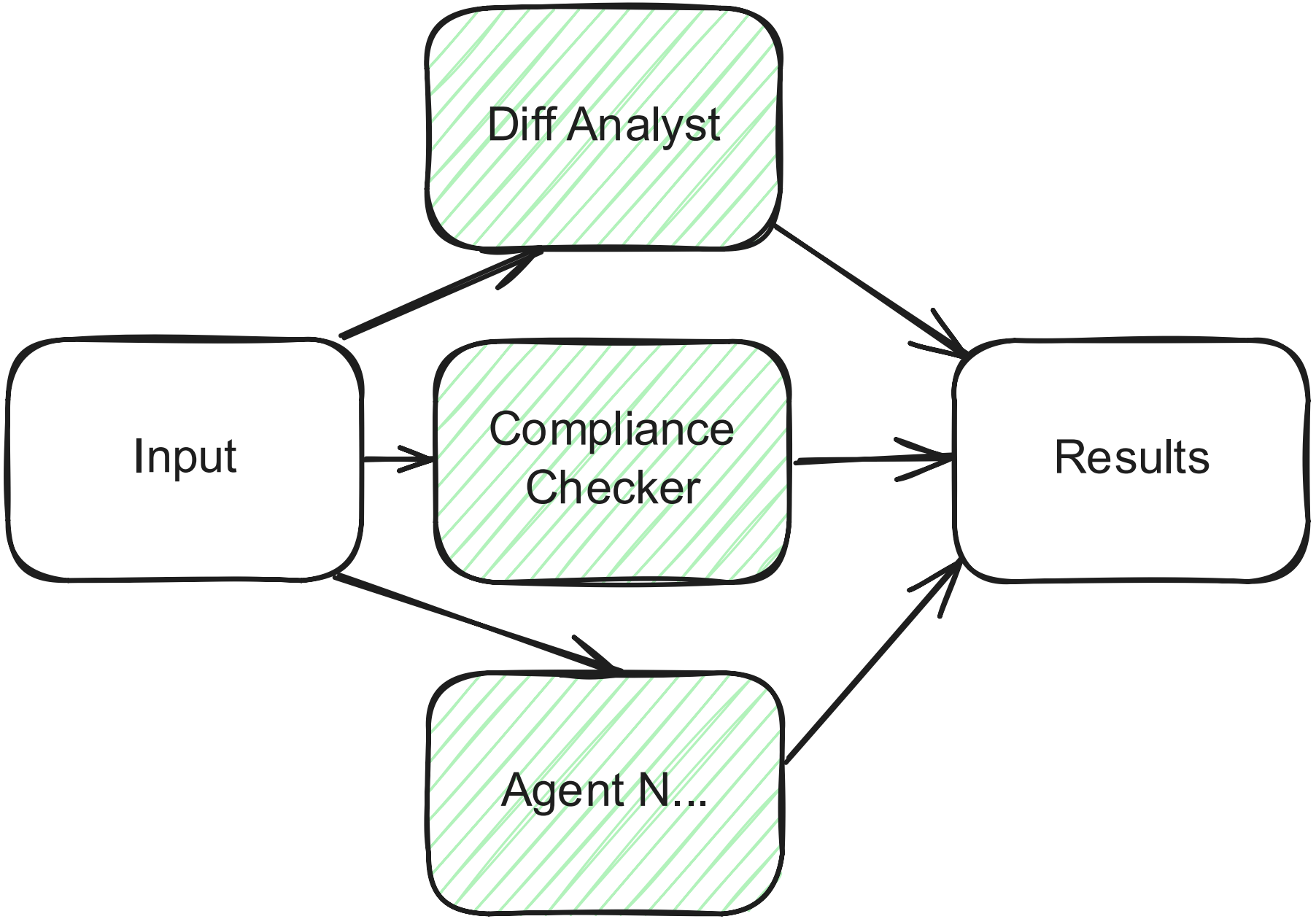
ZURE

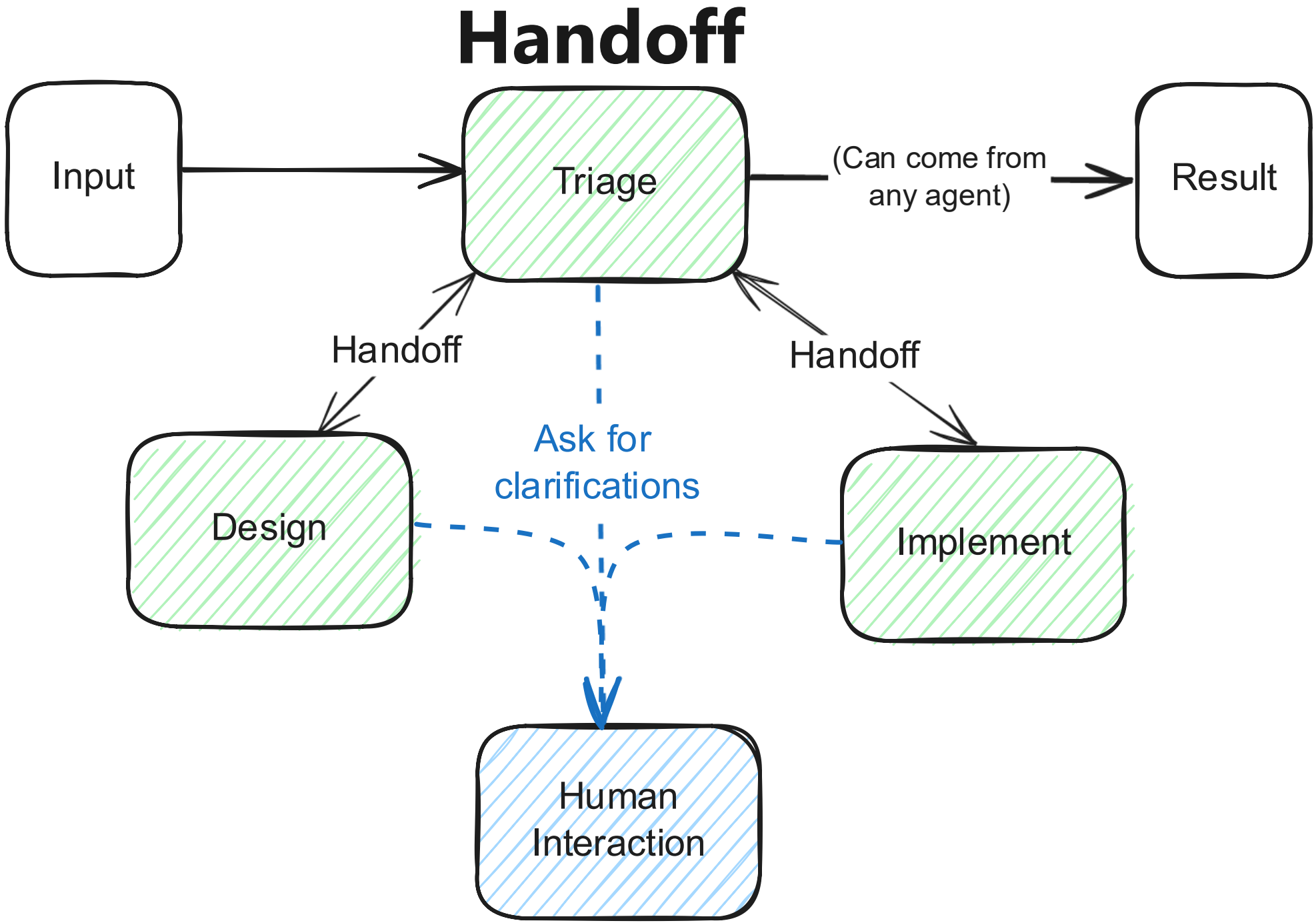
Orchestration Patterns

Sequential

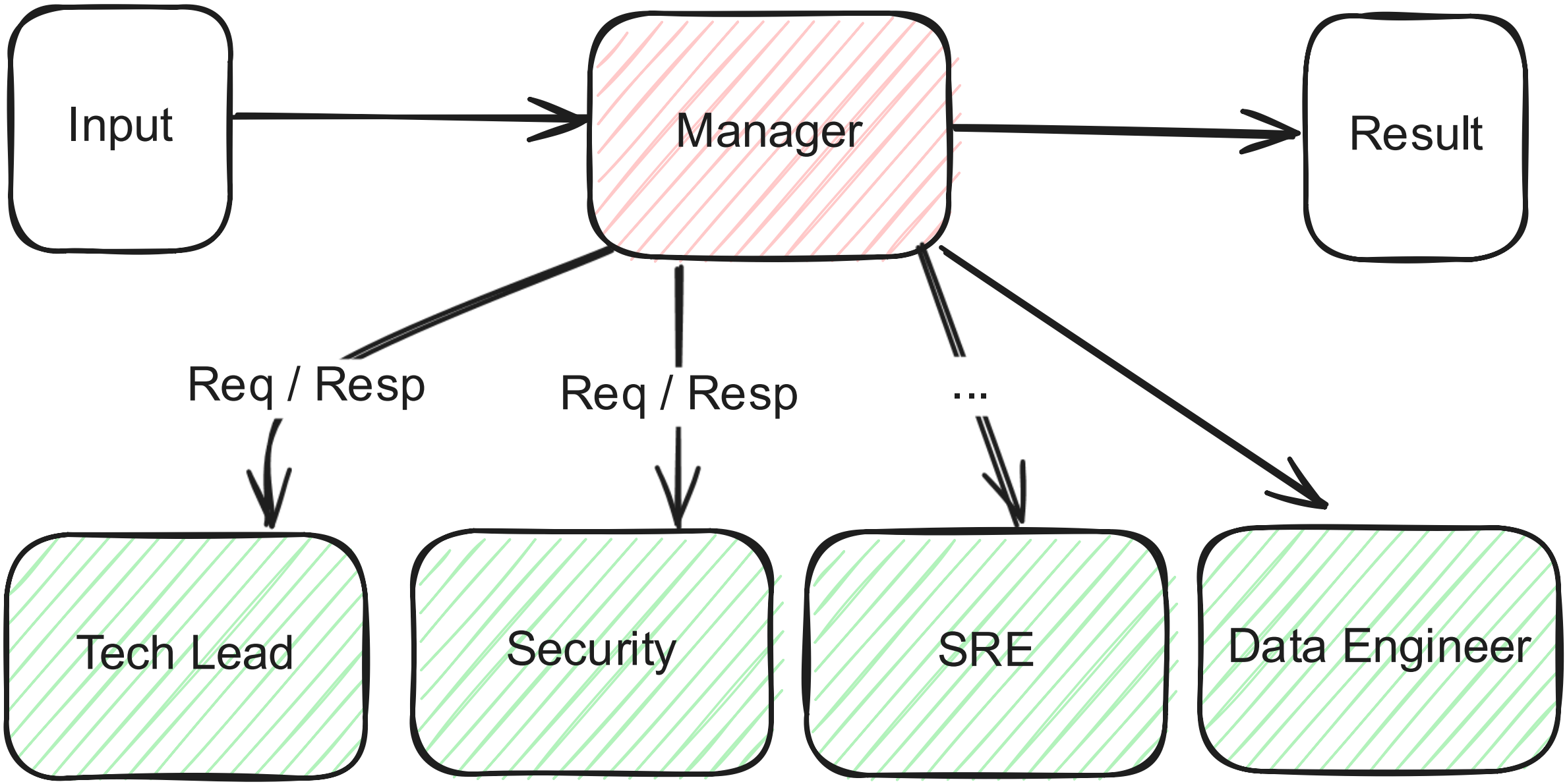

Backlog Refiner
Scaffolder
API Designer
Test Writer
Doc Writer

Concurrent

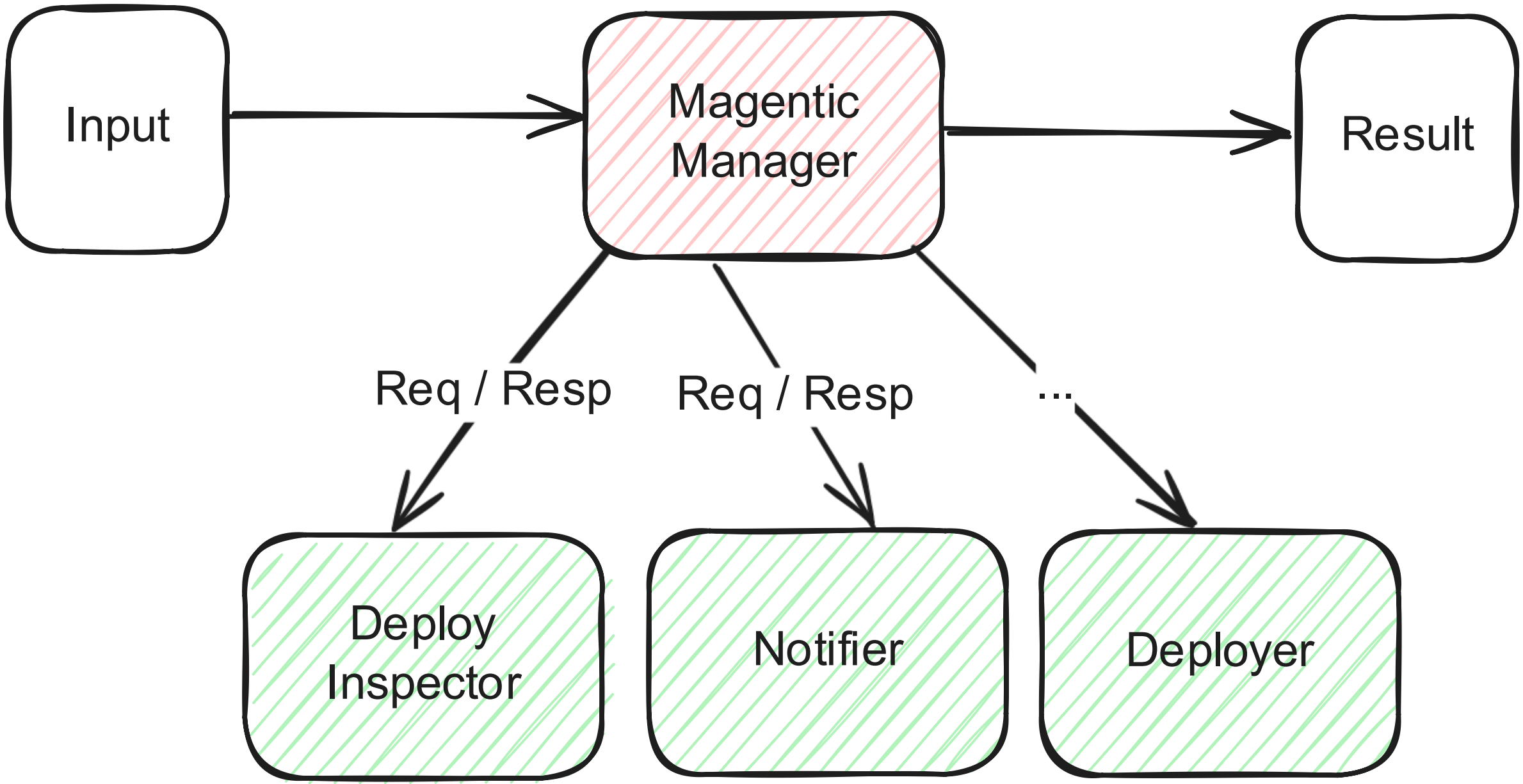


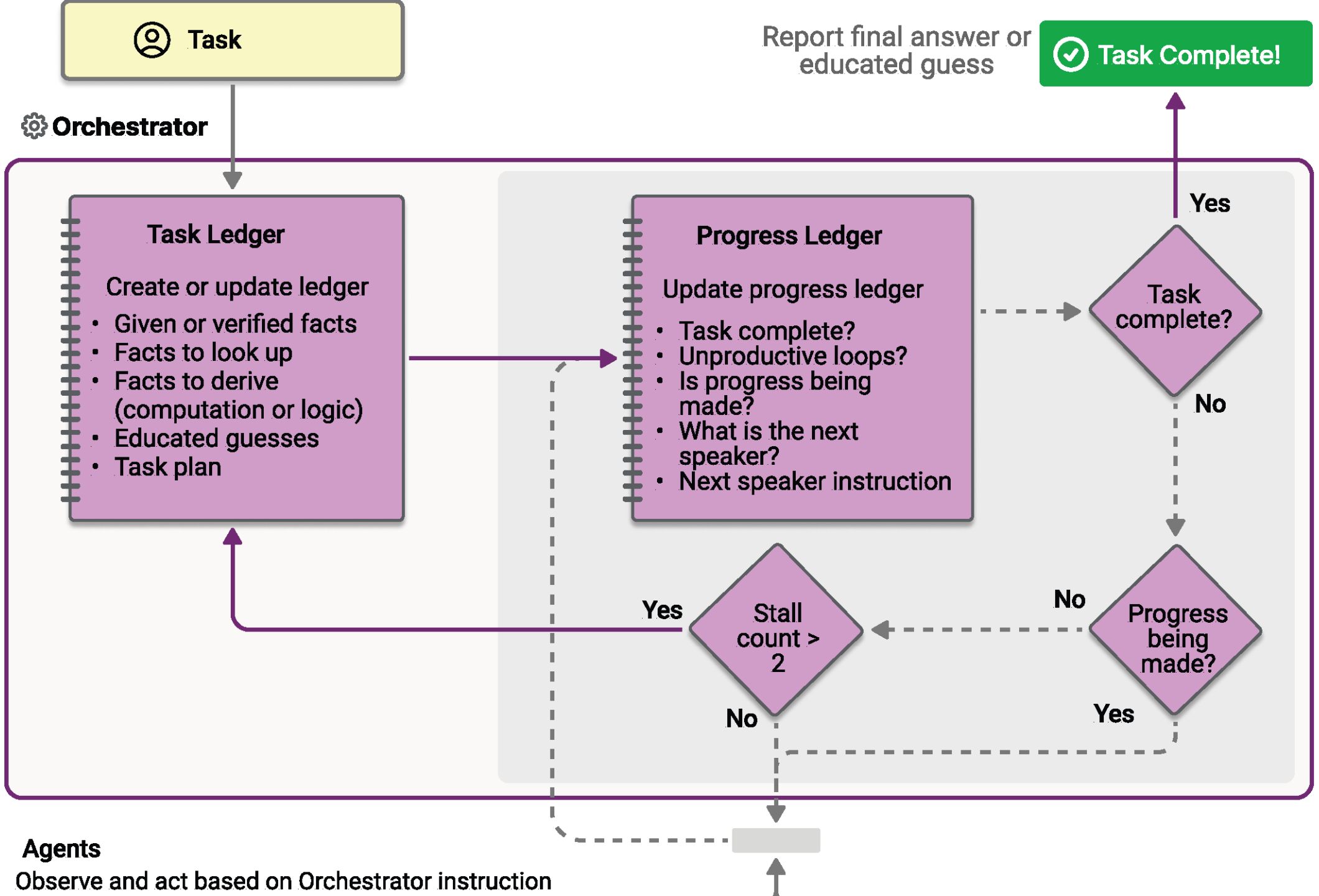

Handoff
Input
Triage
(Can come from any agent)
Result
Handoff
Handoff
Ask for clarifications
Design
Implement
Human Interaction

Group Chat


Input
Manager
Result
Req / Resp
Req / Resp
...
Tech Lead
Security
SRE
Data Engineer

Magentic Orchestration




Task
Orchestrator
Report final answer or educated guess
Task Complete!
Task Ledger
Create or update ledger
• Given or verified facts
• Facts to look up
• Facts to derive (computation or logic)
• Educated guesses
• Task plan
Progress Ledger
Update progress ledger
• Task complete?
• Unproductive loops?
• Is progress being made?
• What is the next speaker?
• Next speaker instruction
Task complete?
Yes
No
Progress being made?
No
Yes
Stall count > 2
Yes
No
Agents
Observe and act based on Orchestrator instruction

Code

ZURE


ZURE