
AGENTIC AI:
FROM BUSINESS NEEDS
TO WORKING AGENTS
Professor Peyman Teymoori
BUSINESS NEEDS
WORKING AGENTS

Learning Objectives

1

DEFINE

What makes software **"agentic"** and identify appropriate use cases

2

ARTICULATE

The **perceive → plan → act loop** and explain when it breaks down

3

TRANSLATE

A business problem into **measurable KPIs** and constraints

4

DESIGN

System instructions that encode **business logic and guardrails**

5

CREATE

Specific, **testable success criteria** for agent behavior

6

IDENTIFY

Common risks (hallucination, cost, privacy) and **mitigation strategies**

What Makes Software "Agentic"? 🎯

Traditional Software

```
Input → Fixed Logic → Output
```

- Follows predetermined rules
- Executes exact commands
- No decision making

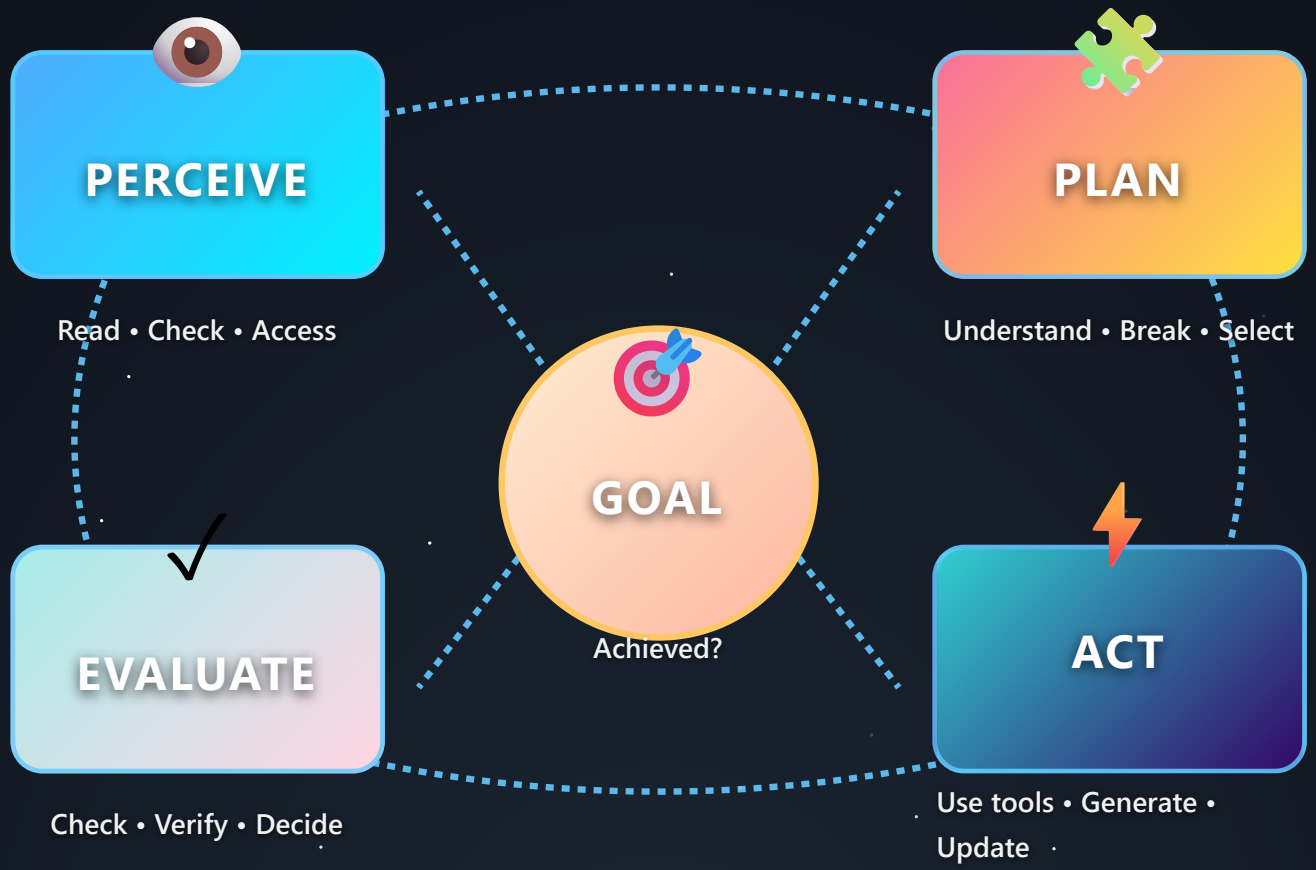
VS

Agentic Software

```
Goal → Perceive → Plan → Act → Evaluate
```

- **Pursues goals** rather than commands
- **Perceives** its environment
- **Makes decisions** autonomously
- **Uses tools** to affect environment
- **Learns from feedback** within session

The Core Agent Loop



Components: Tools

What are Tools?

Tools are **capabilities** the agent can invoke to interact with the world. They're the agent's hands and eyes.

`search_database(query)`

Returns records from database

`send_email(to, subject, body)`

Sends message to user

`calculate(expression)`

Performs mathematical calculation

`retrieve_policy(topic)`

Returns relevant document chunks

Tool Anatomy

```
def tool_name(parameter1: type, parameter2: type) -
> return_type: """ Purpose: What this tool does
When to use: Specific situations Input: Description
of parameters Output: What it returns """ #
Implementation return result
```

Key Principles

- Clear, descriptive names
- Well-defined input schemas
- Predictable output formats
- Good documentation

Components: Memory

Working Memory

Current task only

Steps taken so far, intermediate results, current context

Session Memory

Current conversation

User preferences, context from earlier in chat, conversation flow

Long-term Memory

Across sessions

Historical patterns, learned preferences, user profile

Example Memory Structure

```
{ "user_id": "alice@company.com", "preferences": { "urgency_threshold": "high", "language": "en" }, "history": [ "requested PTO policy", "asked about benefits" ], "context": { "current_task": "schedule_meeting", "step": 2 } }
```

📜 System Instructions: The Agent's Constitution

"The core instructions that define behavior and boundaries"

🪜 Structure

ROLE

Specific role and purpose

GOAL

Primary objective

CAPABILITIES

Available tools and their purposes

CONSTRAINTS

Must do / Must not do

PROCESS

Step-by-step workflow

OUTPUT FORMAT

How to present results

💼 Example: Procurement Agent

ROLE:

Procurement review assistant that validates vendor quotes

GOAL:

Identify potential issues and flag items for human review

CAPABILITIES:

- compare_to_history()
- check_vendor_status()
- calculate_variance()

CONSTRAINTS:

MUST flag quotes >$10,000

MUST NOT auto-approve unapproved vendors

When data missing → "NEEDS_REVIEW"

Example: Email Triage Agent in Action

From: user@company.com

"My laptop won't turn on. URGENT!"

1

PERCEIVE

- Read email content
- Check user history
- Access IT policy

```
Memory: 3 prev tickets
Context: Hardware → L2
```

2

PLAN

- Classify urgency
- Identify category
- Determine routing

```
Urgency: HIGH
Category: HARDWARE
```

3

ACT

- Use classification tool
- Lookup policy
- Generate response

```
classify_ticket() ✓
draft_reply() ✓
```

4

EVALUATE

- Verify classification
- Check compliance
- Confirm routing

```
Status: VERIFIED ✓
Route: L2_SUPPORT
```

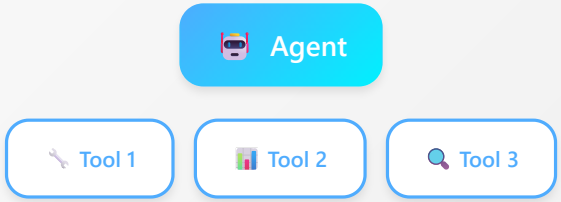
Agent Output

TICKET STATUS	RESPONSE TIME	DRAFT REPLY
Escalated to L2	1.2 seconds	Auto-generated ✓

Single vs Multi-Agent Systems

Single Agent

One agent, multiple tools



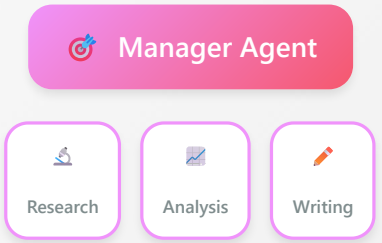
Simple Architecture
Direct control flow, easier to debug

Lower Latency
No coordination overhead

Cost Effective
Single LLM call per decision

Multi-Agent

Specialized delegation system



Specialization
Each agent optimized for tasks

Parallel Processing
Multiple tasks simultaneously

Complex Workflows
Multi-step, branching processes

Industry Use Cases

HR Support Agent

Employee Services

Answers employee questions about policies, benefits, and procedures using company handbook.

- ✓ 24/7 availability
- ✓ Consistent answers
- ✓ Reduces HR workload by 60%

Financial Analyst

Data Analysis

Analyzes financial data, generates reports, and flags anomalies for human review.

- ✓ Faster report generation
- ✓ Early anomaly detection
- ✓ Standardized formats

Customer Service

Support Automation

Handles order inquiries, tracks shipments, and resolves common customer issues.

- ✓ Instant responses
- ✓ Multi-language support
- ✓ Escalates complex cases

Legal Research

Document Analysis

Searches case law, summarizes precedents, and identifies relevant statutes.

- ✓ Hours saved per case
- ✓ Comprehensive coverage
- ✓ Citation accuracy

Agent Canvas

"One-page business case for your agent"

Problem & Value

- What problem?
- For whom?
- Why does it matter?

Target User

- Who interacts?
- User context?
- Technical level?

Success KPIs

- Measurable metrics
- How measured?
- Target values?

Inputs & Outputs

- What comes in?
- What goes out?
- Format specs?

Data Sources

- What data needed?
- Access method?
- Update frequency?

Constraints

- Privacy requirements?
- Latency limits?
- Budget constraints?

Agent Spec Sheet

"Technical blueprint for implementation"

System Instruction

ROLE

Clear definition of agent's purpose and identity

GOAL

Primary objective the agent pursues

CAPABILITIES

List of available tools and when to use them

CONSTRAINTS

→ MUST do / MUST NOT do

→ Safety guardrails

→ Fallback behaviors

Implementation Details

TOOLS

→ Input/output schemas

→ Example usage

→ Error handling

MEMORY

→ What to store

→ When to store

→ Retention policy

EVALUATION

→ Key metrics

→ Pass/fail thresholds

→ Test cases (min 5)

Testing Your Agent

Happy Path

Standard scenarios where everything works as expected

"Employee asks about PTO policy → Agent returns correct policy section"

Edge Cases

Unusual inputs or boundary conditions

"Query with misspellings → Agent still understands intent"

Guardrails

Testing safety boundaries and constraints

"Request to access unauthorized data → Agent refuses appropriately"

Error Handling

How agent behaves when things go wrong

"Database unavailable → Agent provides helpful error message"

Best Practices

- ✓ Clear pass criteria for each test
- ✓ Document expected vs actual behavior
- ✓ Automate repetitive test cases
- ✓ Test with real user data
- ✓ Minimum 5 tests before deployment
- ✓ Include stakeholders in test review

⚠️ Risks & Guardrails

Hallucination

Agent confidently states incorrect information

Mitigation:
Require citations, verify against sources, confidence thresholds

Privacy Leaks

Agent exposes sensitive or personal data

Mitigation:
Access controls, data filtering, audit logs, PII detection

Cost Overruns

Unexpected API calls or infinite loops

Mitigation:
Rate limits, max iterations, budget alerts, caching

Essential Guardrails

Timeouts
Max execution time per task

Iteration Limits
Max steps in agent loop

Human-in-the-Loop
Required approval gates

Audit Logging
Track all agent actions

Scope Constraints
Clear boundaries of authority

Output Validation
Verify format and content

🎯 Workshop Activity

"Design Your Own Agent"

1 Brainstorm

5 minutes

Identify a business problem in your organization that could benefit from an agent

2 Design

10 minutes

Complete the Agent Canvas and draft your Agent Spec Sheet

3 Test

5 minutes

Create 5 test cases covering happy path, edge cases, and guardrails

📦 Your Deliverables

Agent Canvas
One-page business case

Agent Spec
Technical blueprint

5 Tests
Success criteria

👥 Team Roles (Rotating Each Session)

Product Owner
(PO)
Ensure technical choices align with business goals

Agent Architect
(AA)
Lead architecture pattern selection and workflow design

Toolsmith
(TS)
Implement tools and establish data connections

Evaluator
(EV)
Build test workflow and track performance metrics

⭐ Key Takeaways ⭐

1

🎯

Agents are goal-driven

They perceive, plan, and act autonomously—not just execute commands

2

🔄

The Loop is Everything

Perceive → Plan → Act → Evaluate. Master this pattern

3

📜

Instructions = Constitution

System instructions encode business logic, safety rules, and boundaries

4

🛡️

Guardrails are Essential

Safety isn't optional. Build constraints and human gates from day one

5

🧪

Test or Don't Trust

Define clear, measurable success criteria before deployment

6

🚀

Start Small, Scale Smart

Better to do one thing really well than many things poorly

🚀 Next Steps & Lecture 2 Preview

Lecture 2: Patterns, Tools & Memory

From Design → Implementation

🎯 **Architecture Patterns**

ReAct, Plan-and-Execute, Chain-of-Thought

🔧 **Implementing Tools**

Build your first real agent tools

🧠 **Adding Memory**

Context and state management

✅ **Running Evaluations**

Test and validate your agent

Before Next Lecture

- ✓ Incorporate peer feedback into your Canvas and Spec
- ✓ Validate one key assumption (data availability, tool feasibility)
- ✓ Prepare questions about patterns and architecture