TEACHER-FACING FRIDAY DESIGN

Designing Workflows: From Single Chat to Reliable AI Systems

CAS Human-AI Collaboration | Module 1 | Unit 8

LITERACY	COLLABORATION MOVE	CONTACT TIME	CHALLENGE
System	**Orchestration**	**365 minutes**	**Workflow Control**

Position in Module

- **Unit:** 8
- **Topic:** Designing Workflows: From Single Chat to Reliable AI Systems
- **Literacy perspective:** System
- **Collaboration level:** Orchestration
- **Fundamentals focus:** Agentic AI Fundamentals
- **Why this Friday matters:** Participants learn to move beyond a sequence of chat turns and design a controllable agentic system with explicit state, capabilities, routing, validation, permissions, stop conditions, and human approval. They learn when orchestration improves reliability and when it only adds complexity.

Literacy Perspective

- **Primary literacy:** System
- **Secondary literacy, if any:** None
- **What this literacy explains:** System Literacy explains how model calls, tools, resources, skills, prompts, memory, state transitions, permissions, validation, retries, and human checkpoints interact over time. It makes reliability and accountability properties of an architecture rather than of one response.
- **What participants should manipulate, observe, and interpret:** Participants keep the task and input cases stable while manipulating workflow architecture and control points. They observe state continuity, tool selection, validation, failure recovery, approval behaviour, traceability, and unnecessary complexity, then interpret whether orchestration creates sufficient control to justify itself.

Collaboration Level

- **Level relation:** Orchestration
- **Collaboration move being learned:** Participants move from co-constructing or reflecting within bounded interactions to orchestrating work across model calls, tools, resources, roles, state, and human decisions.

- **What participants should be able to do differently after this Friday:** They should be able to map required capabilities and permissions, define a stateful workflow with explicit transitions and stop conditions, place validation and approval points, test normal and failure cases, and decide whether to implement, simplify, or reject the workflow.

Agentic AI Fundamentals

This Friday should introduce agentic AI as the system-level extension of generative AI. The goal is for participants to understand the building blocks needed to implement an AI workflow in a tool such as `n8n`, Google Vertex, or an equivalent environment.

- **Explain the basic agentic loop:** perceive relevant input or state, reason about the task, plan next steps, use tools or resources, observe results, reflect on whether the result satisfies the goal, and update the working state or memory where appropriate.
- **Introduce skills as reusable capability packages:** instructions, procedures, examples, constraints, or domain conventions that allow an agent to perform a recurring class of work more reliably than with an ad hoc prompt.
- Introduce MCP-style capabilities as a useful conceptual model for agentic infrastructure: tools allow actions, resources provide accessible context, and prompts provide reusable interaction patterns. The point is the architecture of capabilities, not dependence on one protocol implementation.
- **Explain workflow state:** outputs from one step become inputs to later steps, and artefacts such as logs, intermediate files, retrieved resources, tool results, and approval records make that state inspectable.
- Show why control points matter. Validation, approval, logging, retries, routing, permissions, and stop conditions prevent the workflow from silently turning uncertain generation into consequential action.
- **Caveat to demonstrate:** tool access increases capability and risk at the same time. An agent with broad permissions can act on a mistaken plan, call the wrong tool, leak context, overwrite useful work, or create cascading errors, so permission boundaries and dry-run modes should be observable parts of the workflow.
- **Emphasize the central learning point for Unit 8:** an agentic workflow is controlled through its architecture of capabilities, state, tools, and checkpoints, not only through the wording of individual prompts.

Recurring Agentic Coding Demonstration

Use one bounded development task that requires repository inspection, a code change, tests, and approval before integration. In the baseline, provide the task to a chat-style assistant that can advise but has no explicit state model or tool workflow. In the target condition, show an agent using durable project instructions, a plan artefact, file and test tools, a reusable skill or procedure, restricted permissions, validation, and an approval gate. Preserve the task, repository snapshot, acceptance criteria, and test cases. Capture the capability map, state transitions, tool calls, intermediate files, validation result, approval record, and final diff. Introduce one benign failure such as a missing dependency or failing test and inspect how the workflow routes, stops, or recovers. Leave adversarial content and prompt-injection analysis to Unit 9.

Shared Classroom Workflow Case

Use one low-risk fictional workflow so participants can inspect system behaviour without connecting to live organisational services.

Default case:

> **A fictional professional-training provider receives workshop enquiries. The system must extract the request, check required fields, consult a scheduling and pricing policy, classify the enquiry, draft a response, and prepare an internal follow-up task. Nothing may be sent or scheduled until a human approves the proposed action.**

Prepare one fixed case pack containing:

- a complete enquiry, an enquiry missing a required field, and an enquiry that conflicts with a policy rule
- a short scheduling and pricing policy stored as an inspectable resource
- an output schema for extracted fields, classification, evidence used, draft response, confidence or uncertainty, and proposed next action
- a single-chat baseline instruction that asks for the complete result in one interaction
- a target workflow skeleton with intake, validation, routing, resource lookup, drafting, output validation, approval, and simulated action steps
- a state schema showing which fields must persist between steps
- a capability-map template covering reusable skills or instructions, tools, resources, prompts, permissions, and human roles
- a trace template for step input, decision, output, tool result, state update, error, and next transition
- one benign failure card, such as unavailable policy resource, malformed field, failed validation, or simulated tool timeout

All external actions should be simulated, sandboxed, or directed to disposable test resources. The Friday is about workflow control, not operational deployment. A group may implement the flow in a workflow platform or execute a tightly specified paper/markdown simulation in which another group plays the tools and environment.

Friday Activity Plan

Indicative contact time is 365 minutes, excluding breaks and lunch. Teachers may adjust durations while preserving baseline capture, architecture design, implementation or simulation, failure testing, comparison, and transfer.

Time	Activity	Concrete output
45 min	From chat sequence to agentic system	Annotated workflow and system-versus-interaction decision notes
45 min	Single-chat baseline	Baseline output, assumption log, and initial scores
45 min	Capability and state architecture	Capability map, state schema, and workflow diagram
65 min	Build or tightly specify the target workflow	Runnable or executable specification plus first complete trace
50 min	Failure injection and recovery	Failure traces, revised control point, and recovery result
45 min	Does orchestration earn its complexity?	Comparison table, simplification decision, and provisional orchestration rule
70 min	Challenge kickoff and first workflow sketch	Challenge protocol, workflow representation, capability map, prediction table, and first trace or test case

Detailed activity blocks

1

45 min

From chat sequence to agentic system

Content and manipulation. Distinguish conversational assistance, Unit 5 process decomposition, and Unit 8 orchestration. Introduce the perceive-plan-act-observe loop, workflow state, capabilities, transitions, and stop conditions.

Method and expected observation. Architecture walkthrough with step-classification exercise; participants identify which elements require a system rather than another prompt.

Concrete output. Annotated workflow and system-versus-interaction decision notes

Tools and materials. Slides, architecture cards, example traces

2

45 min

Single-chat baseline

Content and manipulation. Process the complete workshop enquiry in one chat interaction with the fixed baseline instruction. Preserve the response and note hidden assumptions, missing state, and unverifiable proposed actions.

Method and expected observation. Groups predict likely control failures, run or inspect the baseline, and score it before seeing the target workflow.

Concrete output. Baseline output, assumption log, and initial scores

Tools and materials. Case pack, chat interface or prepared output, rubric

3 45 min

Capability and state architecture

Content and manipulation. Map required skills, prompts, resources, tools, permissions, human roles, data fields, transitions, validation, and approval. Remove capabilities that are unnecessary for the task.

Method and expected observation. Small-group design with least-capability and state-completeness peer check.

Concrete output. Capability map, state schema, and workflow diagram

Tools and materials. Capability cards, state template, workflow canvas

4 65 min

Build or tightly specify the target workflow

Content and manipulation. Implement or simulate intake, required-field validation, routing, policy lookup, drafting, output validation, human approval, and simulated action. Log each state transition.

Method and expected observation. Hands-on build or structured role-play; expected observation is that explicit state and control points make intermediate failures visible.

Concrete output. Runnable or executable specification plus first complete trace

Tools and materials. Workflow tool or markdown simulation kit, test resources

5 50 min

Failure injection and recovery

Content and manipulation. Run the missing-field and policy-conflict cases, then introduce one benign resource, validation, or tool failure. Observe routing, retries, stop behaviour, and escalation.

Method and expected observation. Controlled test exercise; groups may change one control point after the first failure but must record the before/after architecture.

Concrete output. Failure traces, revised control point, and recovery result

Tools and materials. Test-case cards, failure card, trace log

6 45 min

Does orchestration earn its complexity?

Content and manipulation. Compare baseline and workflow on correctness, completeness, state continuity, traceability, approval behaviour, recovery, time, and maintenance burden.

Method and expected observation. Evidence review and simplification challenge; each group identifies one indispensable control and one removable step.

Concrete output. Comparison table, simplification decision, and provisional orchestration rule

Tools and materials. Rubric, baseline and target traces, shared board

7

70 min **Challenge kickoff and first workflow sketch**

Content and manipulation. Select a context-relevant task, define a simple baseline, map capabilities and state, choose one primary orchestration change, and plan normal and failure tests.

Method and expected observation. Individual or group coaching with scope, permission, and feasibility check.

Concrete output. Challenge protocol, workflow representation, capability map, prediction table, and first trace or test case

Tools and materials. Challenge template, workflow canvas, capability map, evidence log

Experiment Focus

- **Experiment focus:** simple single-interaction assistance versus a stateful agentic workflow with explicit capabilities and control points.
- **Primary manipulated condition:** workflow architecture. The target adds defined state, step transitions, resources or tools, validation, trace logging, and at least one human approval or stop condition.
- **Optional focused manipulation:** after the main comparison, vary one control point such as validation, retry logic, permission scope, or approval placement while keeping the rest of the workflow stable.
- **What remains constant across comparison:** Keep the user task, input cases, policy resource, intended output, evaluation criteria, and external-action boundary stable. Record differences in time and implementation effort rather than hiding them.
- **Prediction requirement:** Before testing, predict where the baseline will lose state or oversight and which target control should become visible under a normal, missing-input, or failure case.
- **Action boundary:** Use simulated, sandboxed, read-only, or disposable tools; no live message, booking, deletion, purchase, or consequential update should occur in the classroom experiment.

Observation Focus

- **State continuity:** Are required fields preserved accurately across steps, and are missing values visible rather than silently invented?
- **Capability discipline:** Does each step use only the skill, prompt, resource, tool, and permission needed for its purpose?
- **Control-point activation:** Do validation, stop, retry, routing, and approval points trigger under the cases for which they were designed?
- **Failure recovery:** Does the workflow recover, escalate, or stop predictably when a resource, validation, or simulated tool fails?
- **Traceability:** Can a reviewer reconstruct inputs, decisions, tool results, state changes, and human approvals from the trace?
- **Output quality:** Does the final result satisfy the same requirements more reliably than the baseline?
- **Complexity and accountability:** Which controls justify their cost, which steps add brittle complexity, and who is responsible for approving consequential action?

- **Evidence participants should capture:** baseline interaction and output; workflow diagram or export; capability and permission map; state schema; normal and failure traces; validation and approval records; before/after control-point change; timing; and unexpected behaviour.
- **Interpretation participants should be prepared to make:** Explain which architectural elements improved reliability or accountability, which failure modes remained, whether a simpler design could achieve the same control, and why tool access alone does not make a system agentically mature.

Challenge Kickoff Deliverables

Participants must leave Friday with:

- a selected task that genuinely requires multiple steps, state, a resource or tool, and at least one control point
- a simple baseline interaction or process for comparison
- a target workflow representation naming steps, transitions, state, validation, stop conditions, and human approval
- a capability map covering skills or reusable instructions, tools, resources, prompts, permissions, and responsible human roles
- a prediction table for normal input, missing input, and one benign failure case
- a protocol draft covering the objective, intended output, primary architectural manipulation, constants, evaluation criteria, action boundary, and caveat
- **at least one test case and one first captured artefact:** baseline output, workflow trace, state record, capability map, or peer architecture review
- an implementation decision stating what will be built, what may be tightly specified, and which external actions will remain simulated

The kickoff should already reflect the challenge-focus logic:

- **Difficulty:** Single-chat interaction provides too little reliability, traceability, and oversight for tasks where control matters.
- **Approach:** Implement or tightly specify a stateful multi-step workflow, define its capabilities, permissions, transitions, validation, and human control points, test normal and failure cases, compare it with a simpler baseline, and justify whether orchestration earns its complexity.
- **Artefacts:** protocol document; prediction table; workflow diagram, export, or executable step specification; capability and permission map; state schema; baseline and workflow run log; normal and failure traces; validation and approval records; selected intermediate outputs; concise interpretation; residual-risk statement; and one practical orchestration rule.

Teacher Setup

Mandatory:

- one shared low-risk workflow case with complete, missing-field, and policy-conflict inputs
- an inspectable resource, output schema, baseline instruction, target workflow skeleton, and state schema
- capability-map cards covering skills or reusable instructions, tools, resources, prompts, permissions, and human roles
- a trace format showing step input, decision, output, tool result, state update, error, and next transition
- normal-case, missing-input, and benign failure tests
- a sandbox, mock connector, dry-run mode, or other setup that prevents real external actions
- protocol, prediction, workflow, capability, state, trace, comparison, and evidence-capture templates
- demonstration setup for the recurring agentic coding case, if used
- a paper/markdown workflow simulation so the learning does not depend on a particular platform or successful integration setup

Configurable:

- exact workflow, agent, notebook, or coding environment
- whether the workflow is implemented, partly mocked, or tightly specified and role-played
- capability implementation, including MCP-style tools, resources, and prompts where available
- state persistence method and trace format
- work mode, group size, and role distribution
- exact validation, retry, routing, and approval controls tested
- scaffolding depth and peer-feedback format
- case domain, provided external actions remain low risk and simulated or sandboxed