

The Agent Builder's Blueprint

Design Patterns for Tools, Memory, Context, and Multi-Agent Systems

Ngo Trung Kien

Contents

Copyright and publication notice	1
Introduction: the promise and the boundary	3
Visual frontispiece	4
0.1 Who this book is for	4
0.2 Thesis and falsification condition	4
1 Start With the Work Contract	7
1.1 In plain terms	7
1.1.1 Chapter contract	7
1.2 The decision at stake	7
1.3 Working model	8
1.3.1 See the chapter in practice	8
1.4 Technical deep dive	10
1.4.1 Build sequence	10
1.4.2 Evidence to retain	10
1.4.3 Tests that can disprove the design	10
1.4.4 Design trade-off	10
1.4.5 Chapter-specific operating protocol	10
1.4.6 Operating workflow	13
1.4.7 Evidence and control graph	14
1.5 Worked composite case file — illustrative, not empirical evidence	15
1.5.1 Scenario trace	16
1.5.2 Artifact handoff	17
1.5.3 Evidence application	17
1.5.4 Transfer test	17
1.6 Measurement and recovery decision	17
1.7 Practice lab	18
1.8 Limits and judgment	18
1.9 Takeaways	19
1.10 Reflection	19
2 Select Workflow or Agent Architecture	21
2.1 In plain terms	21
2.1.1 Chapter contract	21
2.2 The decision at stake	21
2.3 Working model	22
2.3.1 See the chapter in practice	22
2.4 Technical deep dive	24
2.4.1 Build sequence	24
2.4.2 Evidence to retain	24

2.4.3	Tests that can disprove the design	24
2.4.4	Design trade-off	24
2.4.5	Chapter-specific operating protocol	24
2.4.6	Operating workflow	27
2.4.7	Evidence and control graph	28
2.5	Worked composite case file — illustrative, not empirical evidence	29
2.5.1	Scenario trace	30
2.5.2	Artifact handoff	31
2.5.3	Evidence application	31
2.5.4	Transfer test	31
2.6	Measurement and recovery decision	32
2.7	Practice lab	32
2.8	Limits and judgment	33
2.9	Takeaways	33
2.10	Reflection	33
3	Specify Typed Tools and Schemas	35
3.1	In plain terms	35
3.1.1	Chapter contract	35
3.2	The decision at stake	35
3.3	Working model	36
3.3.1	See the chapter in practice	36
3.4	Technical deep dive	38
3.4.1	Build sequence	38
3.4.2	Evidence to retain	38
3.4.3	Tests that can disprove the design	38
3.4.4	Design trade-off	38
3.5	Typed tool contract example	38
3.5.1	Chapter-specific operating protocol	39
3.5.2	Operating workflow	41
3.5.3	Evidence and control graph	42
3.6	Worked composite case file — illustrative, not empirical evidence	43
3.6.1	Scenario trace	44
3.6.2	Artifact handoff	45
3.6.3	Evidence application	45
3.6.4	Transfer test	45
3.7	Measurement and recovery decision	45
3.8	Practice lab	46
3.9	Limits and judgment	46
3.10	Takeaways	47
3.11	Reflection	47
4	Engineer Retrieval and Context	49
4.1	In plain terms	49
4.1.1	Chapter contract	49
4.2	The decision at stake	49

4.3	Working model	50
4.3.1	See the chapter in practice	50
4.4	Technical deep dive	52
4.4.1	Build sequence	52
4.4.2	Evidence to retain	52
4.4.3	Tests that can disprove the design	52
4.4.4	Design trade-off	52
4.4.5	Chapter-specific operating protocol	52
4.4.6	Operating workflow	55
4.4.7	Evidence and control graph	56
4.5	Worked composite case file — illustrative, not empirical evidence	57
4.5.1	Scenario trace	58
4.5.2	Artifact handoff	59
4.5.3	Evidence application	59
4.5.4	Transfer test	59
4.6	Measurement and recovery decision	60
4.7	Practice lab	60
4.8	Limits and judgment	61
4.9	Takeaways	61
4.10	Reflection	61
5	Design Memory Lifecycles	63
5.1	In plain terms	63
5.1.1	Chapter contract	63
5.2	The decision at stake	63
5.3	Working model	64
5.3.1	See the chapter in practice	64
5.4	Technical deep dive	66
5.4.1	Build sequence	66
5.4.2	Evidence to retain	66
5.4.3	Tests that can disprove the design	66
5.4.4	Design trade-off	66
5.4.5	Chapter-specific operating protocol	66
5.4.6	Operating workflow	69
5.4.7	Evidence and control graph	70
5.5	Worked composite case file — illustrative, not empirical evidence	71
5.5.1	Scenario trace	72
5.5.2	Artifact handoff	73
5.5.3	Evidence application	73
5.5.4	Transfer test	73
5.6	Measurement and recovery decision	73
5.7	Practice lab	74
5.8	Limits and judgment	74
5.9	Takeaways	75
5.10	Reflection	75

6	Build the Runtime Loop	77
6.1	In plain terms	77
6.1.1	Chapter contract	77
6.2	The decision at stake	77
6.3	Working model	78
6.3.1	See the chapter in practice	78
6.4	Technical deep dive	80
6.4.1	Build sequence	80
6.4.2	Evidence to retain	80
6.4.3	Tests that can disprove the design	80
6.4.4	Design trade-off	80
6.4.5	Chapter-specific operating protocol	80
6.4.6	Operating workflow	83
6.4.7	Evidence and control graph	84
6.5	Worked composite case file — illustrative, not empirical evidence	85
6.5.1	Scenario trace	86
6.5.2	Artifact handoff	87
6.5.3	Evidence application	87
6.5.4	Transfer test	87
6.6	Measurement and recovery decision	87
6.7	Practice lab	88
6.8	Limits and judgment	88
6.9	Takeaways	89
6.10	Reflection	89
	Midpoint visual synthesis	90
7	Add State Checkpoints and Idempotency	91
7.1	In plain terms	91
7.1.1	Chapter contract	91
7.2	The decision at stake	91
7.3	Working model	92
7.3.1	See the chapter in practice	92
7.4	Technical deep dive	94
7.4.1	Build sequence	94
7.4.2	Evidence to retain	94
7.4.3	Tests that can disprove the design	94
7.4.4	Design trade-off	94
7.4.5	Chapter-specific operating protocol	94
7.4.6	Operating workflow	97
7.4.7	Evidence and control graph	98
7.5	Worked composite case file — illustrative, not empirical evidence	99
7.5.1	Scenario trace	100
7.5.2	Artifact handoff	101
7.5.3	Evidence application	101
7.5.4	Transfer test	101
7.6	Measurement and recovery decision	101

7.7	Practice lab	102
7.8	Limits and judgment	102
7.9	Takeaways	103
7.10	Reflection	103
8	Orchestrate Specialists	105
8.1	In plain terms	105
8.1.1	Chapter contract	105
8.2	The decision at stake	105
8.3	Working model	106
8.3.1	See the chapter in practice	106
8.4	Technical deep dive	108
8.4.1	Build sequence	108
8.4.2	Evidence to retain	108
8.4.3	Tests that can disprove the design	108
8.4.4	Design trade-off	108
8.4.5	Chapter-specific operating protocol	108
8.4.6	Operating workflow	111
8.4.7	Evidence and control graph	112
8.5	Worked composite case file — illustrative, not empirical evidence	113
8.5.1	Scenario trace	114
8.5.2	Artifact handoff	115
8.5.3	Evidence application	115
8.5.4	Transfer test	115
8.6	Measurement and recovery decision	115
8.7	Practice lab	116
8.8	Limits and judgment	116
8.9	Takeaways	117
8.10	Reflection	117
9	Connect MCP Safely	119
9.1	In plain terms	119
9.1.1	Chapter contract	119
9.2	The decision at stake	119
9.3	Working model	120
9.3.1	See the chapter in practice	120
9.4	Technical deep dive	122
9.4.1	Build sequence	122
9.4.2	Evidence to retain	122
9.4.3	Tests that can disprove the design	122
9.4.4	Design trade-off	122
9.5	MCP call-and-result inspection example	122
9.5.1	Chapter-specific operating protocol	123
9.5.2	Operating workflow	125
9.5.3	Evidence and control graph	126
9.6	Worked composite case file — illustrative, not empirical evidence	127

9.6.1	Scenario trace	128
9.6.2	Artifact handoff	129
9.6.3	Evidence application	129
9.6.4	Transfer test	129
9.7	Measurement and recovery decision	129
9.8	Practice lab	130
9.9	Limits and judgment	130
9.10	Takeaways	131
9.11	Reflection	131
10	Connect Agents With A2A	133
10.1	In plain terms	133
10.1.1	Chapter contract	133
10.2	The decision at stake	133
10.3	Working model	134
10.3.1	See the chapter in practice	134
10.4	Technical deep dive	136
10.4.1	Build sequence	136
10.4.2	Evidence to retain	136
10.4.3	Tests that can disprove the design	136
10.4.4	Design trade-off	136
10.4.5	Chapter-specific operating protocol	136
10.4.6	Operating workflow	139
10.4.7	Evidence and control graph	140
10.5	Worked composite case file — illustrative, not empirical evidence	141
10.5.1	Scenario trace	142
10.5.2	Artifact handoff	143
10.5.3	Evidence application	143
10.5.4	Transfer test	143
10.6	Measurement and recovery decision	144
10.7	Practice lab	144
10.8	Limits and judgment	145
10.9	Takeaways	145
10.10	Reflection	145
11	Control Cost Latency and Concurrency	147
11.1	In plain terms	147
11.1.1	Chapter contract	147
11.2	The decision at stake	147
11.3	Working model	148
11.3.1	See the chapter in practice	148
11.4	Technical deep dive	150
11.4.1	Build sequence	150
11.4.2	Evidence to retain	150
11.4.3	Tests that can disprove the design	150
11.4.4	Design trade-off	150

11.4.5	Chapter-specific operating protocol	150
11.4.6	Operating workflow	153
11.4.7	Evidence and control graph	154
11.5	Worked composite case file — illustrative, not empirical evidence	155
11.5.1	Scenario trace	156
11.5.2	Artifact handoff	157
11.5.3	Evidence application	157
11.5.4	Transfer test	157
11.6	Measurement and recovery decision	157
11.7	Practice lab	158
11.8	Limits and judgment	158
11.9	Takeaways	159
11.10	Reflection	159
12	Produce the Architecture Decision Record	161
12.1	In plain terms	161
12.1.1	Chapter contract	161
12.2	The decision at stake	161
12.3	Working model	162
12.3.1	See the chapter in practice	162
12.4	Technical deep dive	164
12.4.1	Build sequence	164
12.4.2	Evidence to retain	164
12.4.3	Tests that can disprove the design	164
12.4.4	Design trade-off	164
12.4.5	Chapter-specific operating protocol	164
12.4.6	Operating workflow	167
12.4.7	Evidence and control graph	168
12.5	Worked composite case file — illustrative, not empirical evidence	169
12.5.1	Scenario trace	170
12.5.2	Artifact handoff	171
12.5.3	Evidence application	171
12.5.4	Transfer test	171
12.6	Measurement and recovery decision	171
12.7	Practice lab	172
12.8	Limits and judgment	172
12.9	Takeaways	173
12.10	Reflection	173
	Closing: keep the evidence visible	175
	A compact implementation workbook	177
	Decision register	178
	Evidence scorecard	179
	Exception log	180
	Ninety-day plan	181

Control and test matrix	182
Workflow map	183
13 References	185

Start With the Work Contract

In plain terms

The first interface is the agreement among user owner and system about inputs outcomes constraints evidence and escalation. In an engineered agent runtime, the practical question is **What must the agent runtime accomplish before architecture begins?** For AI application engineers, software architects, platform engineers, and technical leads building production agent systems, the answer counts only when **Draft a work contract and map every field to an observable runtime event** produces reviewable evidence for **contract ambiguity count** without normalizing **starting with framework selection instead of task semantics**.

Chapter contract

DECIDE

What must the agent runtime accomplish before architecture begins?

BUILD

Draft a work contract and map every field to an observable runtime event.

MEASURE

contract ambiguity count

STOP OR REDESIGN WHEN

starting with framework selection instead of task semantics

The decision at stake

For **Start With the Work Contract**, the technical lead with service ownership must decide **what must the agent runtime accomplish before architecture begins?** before the implementation task crosses from discussion into consequence for application engineers, platform operators, and affected users. The minimum decision record is the artifact **Draft a work contract and map every field to an observable runtime event**, the observable **contract ambiguity count**, and the explicit hold condition **starting with framework selection instead of task semantics**. If any Chapter 1 field is missing, the safe default is to disable the affected adapter and restore the last tested runtime.

The bounded **Start With the Work Contract** decision advances the book's promise to turn an agent use case into a modular architecture with explicit contracts, state, interoperability, cost controls, and engineering decisions. It does not endorse a product or predetermine the Chapter 1 answer; it creates a record that another person can challenge at every contract, protocol, state-machine, and release change.

Working model

Read the **Start With the Work Contract** model as four linked fields. **Outcome:** The first interface is the agreement among user owner and system about inputs outcomes constraints evidence and escalation. **Boundary:** the design must make **starting with framework selection instead of task semantics** visible rather than convenient. **Evidence:** interpret **contract ambiguity count** together with review effort and excluded cases. **Authority:** the technical lead with service ownership may continue, narrow, escalate, or stop the implementation task. For **Start With the Work Contract**, Reference [1] from Anthropic, *Building effective agents*, informs a Chapter 1 boundary; it is not evidence that the local method or illustrative target will work.

See the chapter in practice

The **start with the implementation task contract in practice** maps *The Agent Builder's Blueprint* Chapter 1 into a field scene before its abstract models begin. In one frame, it holds the question **What must the agent runtime accomplish before architecture begins?**, the artifact **Draft a work contract and map every field to an observable runtime event.**, the proof signal **contract ambiguity count**, and the warning **starting with framework selection instead of task semantics**.

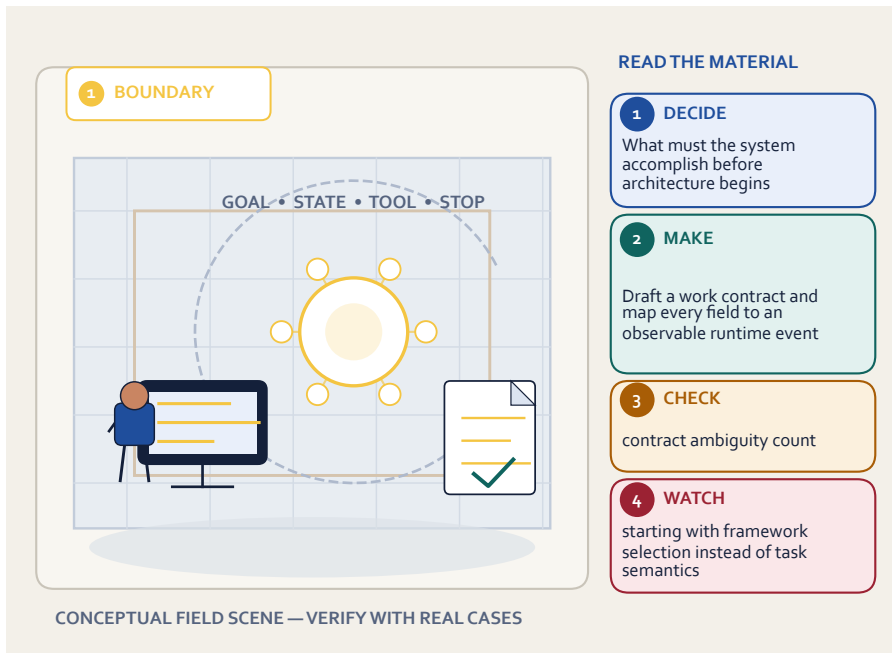


Figure 1.1: Chapter 1 field scene for Start With the Work Contract: a boundary view in the book’s builder setting.

Read the scene: For The Agent Builder’s Blueprint, Chapter 1, begin with the central builder scene and scan DECIDE, MAKE, CHECK, and WATCH. Treat the Start With the Work Contract scene as conceptual, then use its cases and evidence records to test whether contract ambiguity count actually improves. For **Start With the Work Contract**, this explanatory plate is a reasoning prompt—not observed evidence or a universal operating prescription.

Visual field note — Start With the Work Contract:

- 1. Locate the tension.** In this composite situation, A team builds a travel agent before agreeing whether it may reserve or only recommend.
- 2. Name the choice.** Answer the chapter question: **What must the agent runtime accomplish before architecture begins?**
- 3. Produce the record.** Use **Draft a work contract and map every field to an observable runtime event.**, then retain the technical lead, assumptions, and case evidence needed to review it.
- 4. Test and return.** Track **contract ambiguity count**. If **starting with framework selection instead of task semantics**, return to DECIDE and revise the architecture boundary instead of explaining the warning away.

Technical deep dive

The implementation task contract is the domain boundary for the whole system. It defines task input, desired state change, forbidden state changes, required evidence, service limits, accountable owner, and escalation target before code chooses how to act.

Build sequence

1. Interview the task owner using real cases and exceptions.
2. Normalize inputs and outputs into typed domain objects.
3. Write acceptance and refusal examples.
4. Map every contract clause to a log field or test.

Evidence to retain

- Versioned work contract
- Example and counterexample set
- Clause-to-test traceability table

Tests that can disprove the design

- Required input absent
- Outcome achieved through a forbidden action
- Task ownership changes mid-run

Design trade-off

A strict contract improves testability but may exclude legitimate edge cases until the domain model matures. Within *The Agent Builder's Blueprint*, record the rationale for this trade-off, the signal that would reverse it, and the person authorized to decide.

Chapter-specific operating protocol

Phase	Reader action	Evidence retained	Hold condition
1. Define	Write the exact decision behind What must the agent runtime accomplish before architecture begins?	Chapter 1 owner, input, output, exception, and timestamp.	Hold Start With the Work Contract if its record cannot be reconstructed.
2. Contrast	Apply Draft a work contract and map every field to an observable runtime event to one ordinary case.	Chapter 1 owner, input, output, exception, and timestamp.	Hold Start With the Work Contract if its record cannot be reconstructed.
3. Exclude	Run the same artifact against a counter case shaped by starting with framework selection instead of task semantics.	Chapter 1 owner, input, output, exception, and timestamp.	Hold Start With the Work Contract if its record cannot be reconstructed.
4. Assign	Name the technical lead with service ownership as the person who may continue, narrow, or stop.	Chapter 1 owner, input, output, exception, and timestamp.	Hold Start With the Work Contract if its record cannot be reconstructed.
5. Challenge	Record contract ambiguity count and the review effort for the same case.	Chapter 1 owner, input, output, exception, and timestamp.	Hold Start With the Work Contract if its record cannot be reconstructed.

Phase	Reader action	Evidence retained	Hold condition
6. Publish	Release the Start With the Work Contract artifact only when its evidence answers this chapter's question.	Chapter 1 owner, input, output, exception, and timestamp.	Hold Start With the Work Contract if its record cannot be reconstructed.

This sequence is specific to **Start With the Work Contract**. For that Chapter 1 decision, it is not permission to automate; it is a way for AI application engineers, software architects, platform engineers, and technical leads building production agent systems to make the reasoning inspectable at every contract, protocol, state-machine, and release change.

Draft a work contract and map every field to an observable runtime event. This is the chapter's core production artifact. The **Start With the Work Contract** artifact should fit on one page, use ordinary language, and be reviewed by someone who did not design it. A compact artifact exposes disagreements that a long narrative can conceal.

The chapter's visual tool is the **chapter contract**. For **Start With the Work Contract**, it translates explanation into an inspectable decision aid: the reader can see the decision, artifact, evidence, and exact condition that pauses the implementation task.

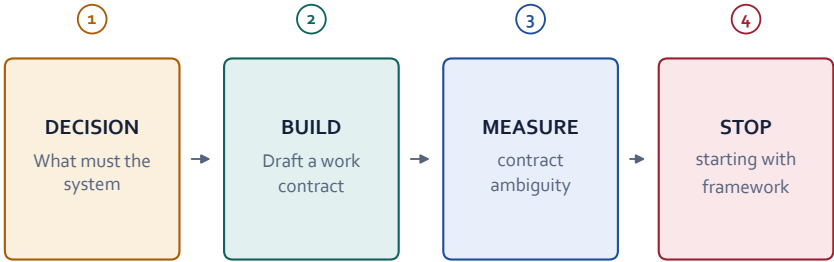


Figure 1.2: Chapter contract connecting the decision, artifact, measure, and stop condition for Start With the Work Contract.

How to use this page: Move across the four Start With the Work Contract cards; compare the decision with contract ambiguity count and stop before starting with framework selection instead of task semantics. The chapter-specific measure is **contract ambiguity count**; the principal warning is **starting with framework selection instead of task semantics**.

Operating workflow

The **start with the implementation task contract workflow** makes the chapter executable. For **Start With the Work Contract**, the five stages carry **Draft a work contract and map every field to an observable runtime event.** from the question **What must the agent runtime accomplish before architecture begins?** to a test against **contract ambiguity count**, with **starting with framework selection instead of task semantics** as an explicit return or stop signal.

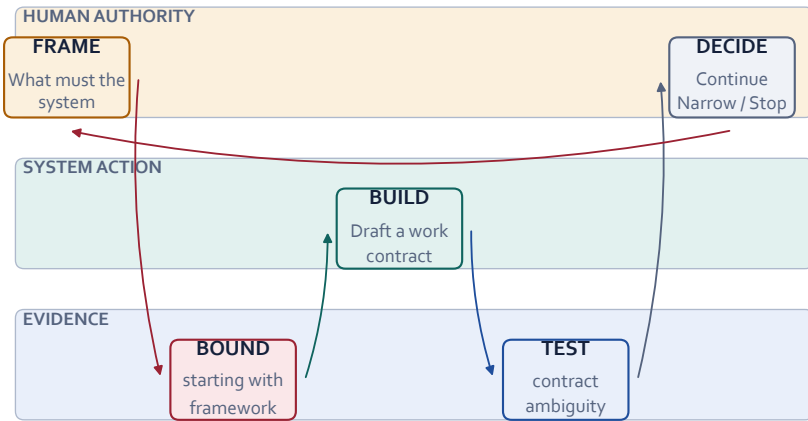


Figure 1.3: Five-stage workflow for chapter 1, Start With the Work Contract: frame the decision, set the architecture boundary, build Draft a work contract and map every field to an observable runtime event., test against contract ambiguity count, then continue, narrow, or stop before starting with framework selection instead of task semantics.

Read the workflow: Move from FRAME to DECIDE for Start With the Work Contract. The lower return path is mandatory when evidence is missing or the stop signal—starting with framework selection instead of task semantics—appears. For **Start With the Work Contract**, record the technical lead and evidence at every handoff; an arrow is not evidence that the preceding stage passed.

Evidence and control graph

The **start with the implementation task contract — evidence network** shows the relationships that must remain inspectable when the workflow runs. In *The Agent Builder's Blueprint*, this qualitative graph asks the reader to record observed evidence for **contract ambiguity count** and route **starting with framework selection instead of task semantics** to recovery instead of treating sample numbers as facts.

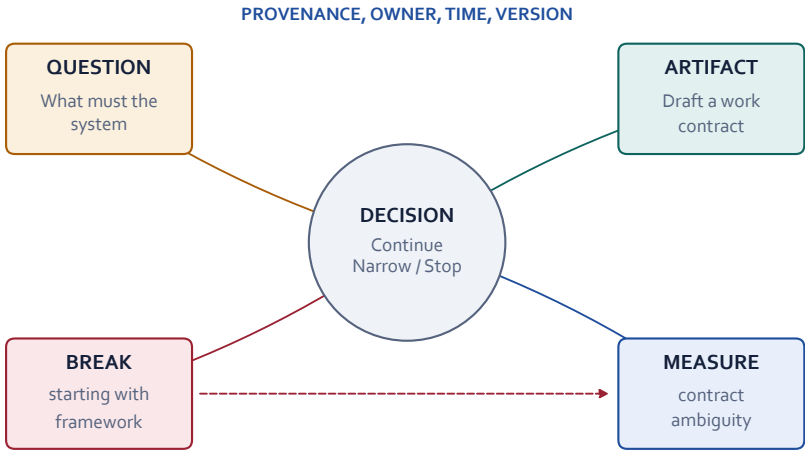


Figure 1.4: Qualitative evidence network for chapter 1, Start With the Work Contract, linking the question to Draft a work contract and map every field to an observable runtime event., the measure contract ambiguity count, and the recovery condition starting with framework selection instead of task semantics without invented empirical values.

Read the graph: Trace the evidence for Start With the Work Contract toward the decision. For Start With the Work Contract, empty or weak evidence does not count as zero risk; it routes the implementation task to verification, escalation, or stop. Use this **Start With the Work Contract** graph beside **Draft a work contract and map every field to an observable runtime event**, not as a substitute for recorded cases.

Worked composite case file — illustrative, not empirical evidence

This **Start With the Work Contract** record turns the scenario into an inspectable rehearsal in an engineered agent runtime. Its numbers are worksheet examples for Chapter 1, not observed results or a prediction of benefit.

Record field	Filled illustrative entry
Situation	A team builds a travel agent before agreeing whether it may reserve or only recommend.
People affected	application engineers, platform operators, and affected users
Decision	What must the agent runtime accomplish before architecture begins?
Artifact excerpt	Draft a work contract and map every field to an observable runtime event.
Accountable authority	The technical lead with service ownership.
Measure and unit	contract ambiguity count — one dated observation for each eligible case.
Example baseline	7 observations in the baseline window.
Example target / decision rule	a documented improvement plus no severe unresolved exception; move only contextual; the technical lead must define better before collection.
Named exception	starting with framework selection instead of task semantics.
Safe default	Disable the affected adapter and restore the last tested runtime.

Scenario trace

1. **Preserve the starting state.** For **Start With the Work Contract**, record the original request, governing rule, available evidence, and the last safe result before changing anything.
2. **Make the chapter decision.** Answer **What must the agent runtime accomplish before architecture begins?** and record the Chapter 1 assumption that carries the most consequence for application engineers, platform operators, and affected users.
3. **Use the artifact.** Complete **Draft a work contract and map every field to an observable runtime event** with a normal case and a counter case; in this chapter, do not infer missing facts merely to finish the form.
4. **Observe the decision rule.** Review **contract ambiguity count** at every contract, protocol, state-machine, and release change; the

Start With the Work Contract record reports exclusions and review effort beside the headline value.

5. **Contain and recover.** If **starting with framework selection instead of task semantics**, disable the affected adapter and restore the last tested runtime. Preserve the Chapter 1 failed case, assign a deadline, and require the technical lead with service ownership to authorize any restart.

Artifact handoff

Start **Start With the Work Contract** from the book-level thesis and boundary. Within *The Agent Builder's Blueprint*, attach the current artifact and exception record, then hand both to Chapter 2, **Select Workflow or Agent Architecture**, which asks: Which parts should be deterministic and which need model-selected control flow?. That recipient may revise the **Start With the Work Contract** assumption, but must preserve the evidence that made this book's earlier decision reasonable.

Evidence application

Reference [1], Anthropic, *Building effective agents*, provides a relevant boundary or design input for **Start With the Work Contract**. It does **not** prove that the Chapter 1 worked values, local method, or target will succeed. The **Start With the Work Contract** reviewer must link this book's exact claim to the exact source passage and leave unsupported synthesis marked UNVERIFIED.

Transfer test

- Re-run **Draft a work contract and map every field to an observable runtime event** with one routine Chapter 1 case that should pass.
- Re-run the **Start With the Work Contract** artifact with missing evidence and keep the safe default active.
- Re-run it where **starting with framework selection instead of task semantics** appears plausible but not certain.
- Ask a reviewer who did not design this chapter's method to decide whether the retained record supports continuing, narrowing, escalating, or stopping.

Measurement and recovery decision

For **Start With the Work Contract**, define **contract ambiguity count** as one dated observation for each eligible case. In this **Start With the Work Contract** worksheet, the Chapter 1 worked baseline is 7 observations in the

baseline window and the illustrative target is a documented improvement plus no severe unresolved exception; both must be replaced with evidence from an engineered agent runtime. Review the Chapter 1 result at every contract, protocol, state-machine, and release change, preserve excluded and high-consequence cases, and interpret movement as better only when it is contextual; the technical lead must define better before collection.

The named failure is **starting with framework selection instead of task semantics**. Containment means: disable the affected adapter and restore the last tested runtime. Recovery is complete only after the technical lead with service ownership can reconstruct the failed case, explain which assumption changed, and authorize a new test against **contract ambiguity count**.

Practice lab

Choose one live but reversible **Start With the Work Contract** example from an engineered agent runtime. Create the following:

- a one-sentence answer to **What must the agent runtime accomplish before architecture begins?**;
- a completed first draft of **Draft a work contract and map every field to an observable runtime event**;
- one routine case, one missing-evidence case, and one case shaped by **starting with framework selection instead of task semantics**;
- a locally measured baseline for **contract ambiguity count** using the unit one dated observation for each eligible case;
- a stop threshold, the technical lead with service ownership as owner, and the safe default disable the affected adapter and restore the last tested runtime;
- architecture evidence packet that will be handed to the next chapter.

Ask a colleague to challenge **Draft a work contract and map every field to an observable runtime event**. They should identify one ambiguous field, one person among application engineers, platform operators, and affected users whose interests are missing, and one reason **contract ambiguity count** could reward the wrong behavior. Preserve that challenge in the chapter handoff before revising.

Limits and judgment

Start With the Work Contract supplies a locally testable method, not proof of causal effectiveness or a universal rule. This Chapter 1 method remains inside the book boundary: This book does not endorse a vendor, guarantee interoperability, or treat a reference architecture as a substitute for workload-specific engineering. Readers applying **Start With the Work Contract** for application engineers, platform operators, and affected users

must verify current law, contract, policy, safeguarding, accessibility, security, and professional duties before consequential use. A negative **contract ambiguity count** result may mean the question was framed badly, the review cost is too high, or disable the affected adapter and restore the last tested runtime is the responsible decision.

Takeaways

- The first interface is the agreement among user owner and system about inputs outcomes constraints evidence and escalation.
- The chapter artifact is: draft a work contract and map every field to an observable runtime event.
- The primary measure is **contract ambiguity count**, paired with review cost and failure severity.
- Watch especially for **starting with framework selection instead of task semantics**.
- In an engineered agent runtime, expand scope only after the technical lead with service ownership can show the evidence and exercise the recovery path.

Reflection

What evidence would let a skeptical reviewer answer **What must the agent runtime accomplish before architecture begins?** differently from the technical lead with service ownership? Which case involving application engineers, platform operators, and affected users would **contract ambiguity count** miss, and would that omission require continuing, narrowing, escalating, or disable the affected adapter and restore the last tested runtime?

