

TIL: Evaluating Tool Use Quality with Google ADK 1.16

Date: 2025-10-21

Difficulty: intermediate

Reading Time: 8 minutes

Tags: til, quick-learn, evaluation, metrics, adk-1.16, tool-use, quality-assurance

Description: Quick guide to using Rubric Based Tool Use Quality Metric: evaluate how well your agents select and use tools to accomplish tasks with ADK 1.16's new evaluation framework

TIL: Evaluating Tool Use Quality - Does Your Agent Pick the Right Tools?

| The Problem

Scenario: Your agent gets the final answer right, but did it get there the right way?

Imagine you have a data analysis agent that:

- ✓ Produces the correct report (final answer is good)
- ✗ But uses the wrong data query tool (inefficient)
- ✗ Calls tools in the wrong order (wasteful)
- ✗ Misses tool combinations that would be better (suboptimal)

The Gap: Evaluating just the final answer misses these critical problems. You need to assess how well the agent used its tools.

In one sentence: Tool Use Quality Metric evaluates whether agents selected the right tools and used them appropriately to solve problems, separate from final answer quality.

| Why Should You Care?

Problems it solves:

- 🎯 **Catches tool selection mistakes** - Agent picks wrong tool even though correct one exists
- ↺ **Prevents tool misuse** - Ensures tools are called in logical sequence
- ⚡ **Identifies inefficiency** - Spots unnecessary tool calls or missing combinations
- 📊 **Separates concerns** - Different metrics for tool quality vs answer quality
- 🔍 **Improves agent training** - Gives precise feedback on tool-related failures

Real-world impact:

A customer support agent that:

- Calls `get_customer_info()` AFTER `process_refund()` (wrong order)
- Gets the refund processed correctly (final answer looks good)
- But violates business logic (should verify customer first)

Without Tool Use Quality metrics: You miss this bug. Final answer was "correct."

With Tool Use Quality metrics: You catch it immediately. The tool sequence evaluation fails.

| Quick Example: Running Real Evaluation

The ADK provides `AgentEvaluator.evaluate()` to run real RUBRIC_BASED_TOOL_USE_QUALITY_V1 evaluation:

```
import asyncio
from google.adk.evaluation.agent_evaluator import AgentEvaluator

async def run_evaluation():
    # Evaluation automatically:
    # 1. Loads your agent module
    # 2. Runs it against test cases in evalset.json
    # 3. LLM judge evaluates tool sequences against rubrics
    # 4. Reports pass/fail with scores

    results = await AgentEvaluator.evaluate(
        agent_module="tool_use_evaluator",
        eval_dataset_file_path_or_dir="tool_use_quality.evalset.json",
    )

    # Results include:
    # - Overall score (0.0-1.0)
    # - Per-rubric scores
    # - Expected vs actual tool calls (side-by-side)
    # - Pass/fail status vs threshold
    print(results)

asyncio.run(run_evaluation())
```

What this does:

1. ✓ Loads `tool_use_evaluator` agent
2. ✓ Reads test cases from evalset (good/bad tool sequences)
3. ✓ Runs evaluation config with 4 rubrics
4. ✓ Calls Gemini model as judge (3 samples per rubric)
5. ✓ Returns scores and detailed comparison

Expected output:

Summary: `EvalStatus.FAILED` for Metric: `rubric_based_tool_use_quality_v1`.
Expected threshold: `0.7`, actual value: `0.25`.

[Detailed table showing:]

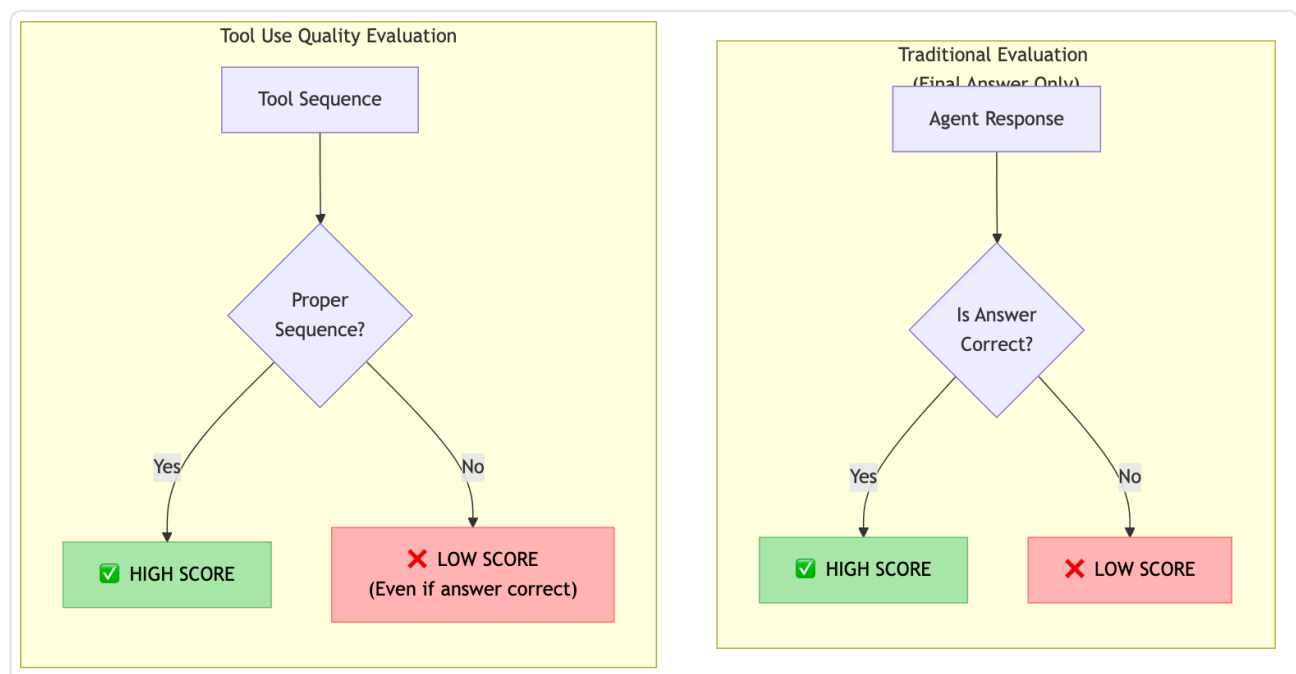
- eval_status: FAILED/PASSED
- score: 0.25 (failed threshold of 0.7)
- prompt: "Analyze and prepare the dataset..."
- expected_response: "Dataset analyzed..."
- actual_response: "Okay, I will analyze the dataset..."
- expected_tool_calls: analyze_data → extract_features → validate_quality
- actual_tool_calls: [empty if not invoked, or list of actual calls]

Key insight: The evaluation shows exactly where agents deviate from expected tool sequences, making it easy to identify and fix tool usage problems.

How It Works (3 Key Concepts)

1. Separating Tool Quality from Answer Quality

Unlike final response evaluation, tool use quality focuses specifically on tool-related decisions:



Key insight: A correct answer from the wrong tool sequence = lower tool use quality, even with high final answer quality.

2. Rubric-Based Evaluation Framework

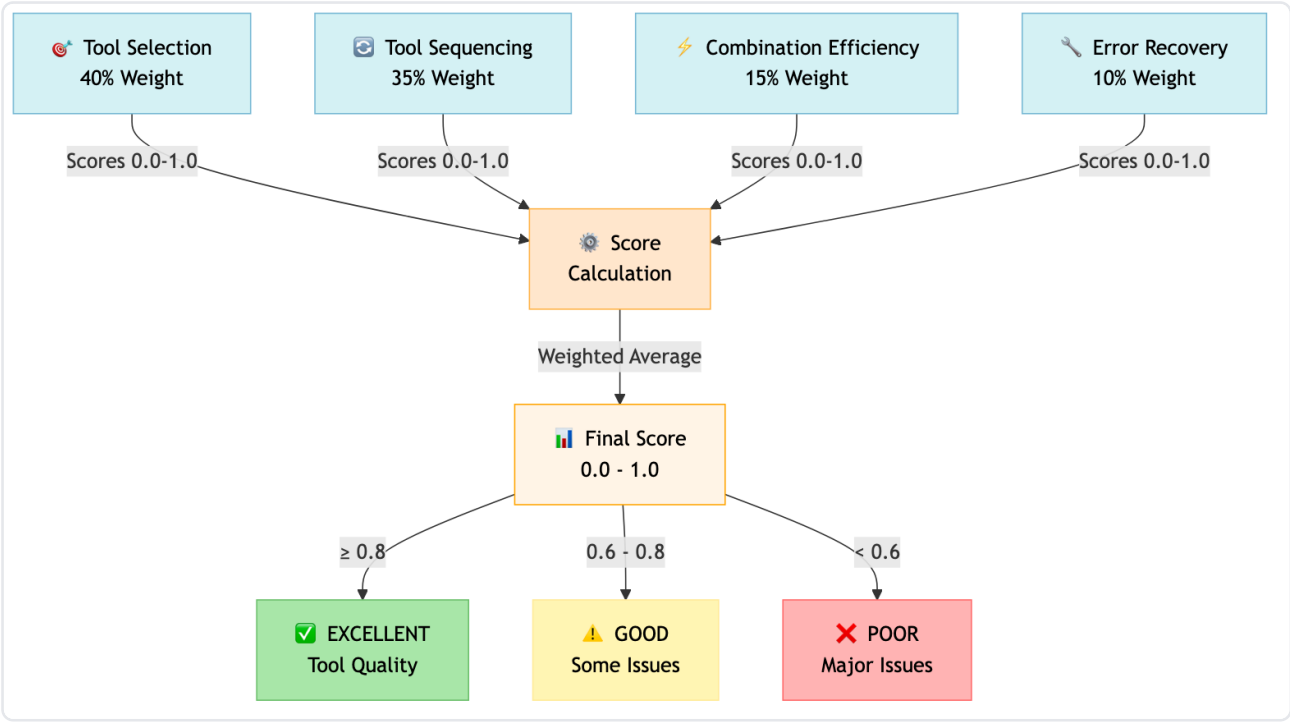
The metric uses a rubric that checks:

TOOL USE QUALITY RUBRIC:

1. Tool Selection Appropriateness (40%)
 - ✓ Were the selected tools suitable for the task?
 - ✗ Did agent use wrong tool when better option existed?
 - ✗ Did agent use deprecated/inefficient tool?
2. Tool Sequencing Logic (35%)
 - ✓ Were tools called in logical order?
 - ✗ Tool 3 depends on Tool 2 output (backwards)?
 - ✗ Unnecessary context switches between tools?
3. Tool Combination Efficiency (15%)
 - ✓ Did agent combine tools effectively?
 - ✗ Missing obvious tool combinations?
 - ✗ Over-using tools when one would suffice?
4. Error Recovery (10%)
 - ✓ If tool failed, did agent retry/alternate?
 - ✗ Gave up after first tool failure?
 - ✗ Repeated same failing tool call?

TOTAL QUALITY SCORE = Weighted average of above

Visual breakdown of how rubrics combine into final score:



3. Distinction from Other Metrics

Metric	Evaluates	Example
Final Response Quality	Is the answer correct?	"Is the report accurate?"
Tool Use Quality	Did agent use tools well?	"Did agent query the right database?"
Safety Evaluator	Is it safe/compliant?	"Did agent violate guardrails?"
Trajectory	Full execution path	"Was the entire reasoning solid?"

Key difference: Tool use is about **PROCESS**, not RESULT.

Example Query: "Get customer's last 3 orders and calculate refund"

AGENT A:

1. Calls: `get_customer()` → Returns customer info
2. Calls: `get_orders()` → Returns last 3 orders
3. Calls: `calculate_refund()` → Correct answer!

AGENT B:

1. Calls: `calculate_refund()` → Error (no customer context)
2. Calls: `get_customer()` → Returns customer info
3. Calls: `get_orders()` → Correct answer finally!

Final Response Quality:

Agent A: ✓ High (correct answer)

Agent B: ✓ High (correct answer)

Tool Use Quality:

Agent A: ✓ High (logical sequence, right tools)

Agent B: ✗ Low (wrong order, wasted call)

Use Case 1: Data Analytics Agent

```
# Evaluate a data analysis agent's tool usage
criterion = Criterion(
    name="data_pipeline_quality",
    description="Agent uses correct data tools in optimal order",
    metric=PrebuiltMetrics.RUBRIC_BASED_TOOL_USE_QUALITY_V1,
    threshold=0.8,
)

# Good tool sequence: Extract → Transform → Aggregate
# Bad tool sequence: Aggregate → Transform → Extract (wrong order!)
# Or: Extract → Extract → Extract (missing Transform step)
```

What it catches:

- ✓ Correct tool order (Extract before Transform)
- ✗ Skipping necessary transformation steps
- ✗ Using wrong data source tool
- ✗ Missing joins between datasets

Use Case 2: Customer Support Agent

```
# Evaluate support agent's tool usage
criterion = Criterion(
    name="support_workflow_quality",
    description="Agent follows proper support workflow",
    metric=PrebuiltMetrics.RUBRIC_BASED_TOOL_USE_QUALITY_V1,
    threshold=0.75,
)

# Proper workflow: Verify → Assess → Resolve
# Bad workflow: Resolve → Verify (backwards!)
# Or: Verify → Verify → Assess (redundant)
```

What it catches:

- ✓ Verification before refund processing
- ✗ Processing refund without checking customer status
- ✗ Checking same condition twice
- ✗ Missing authorization step

Use Case 3: Research Agent

```
# Evaluate research agent's tool usage
criterion = Criterion(
    name="research_methodology",
    description="Agent uses research tools methodically",
    metric=PrebuiltMetrics.RUBRIC_BASED_TOOL_USE_QUALITY_V1,
    threshold=0.85,
)

# Good sequence: Search → Retrieve → Cross-reference → Synthesize
# Bad sequence: Search → Synthesize (missing verification)
# Or: Search → Cross-reference → Retrieve (wrong order)
```

What it catches:

- ✓ Proper research flow
- ✗ Synthesizing before verifying sources
- ✗ Missing comparative analysis steps

-  Not cross-checking information

Configuration Reference

```
from google.adk.evaluation import (
    Criterion,
    PrebuiltMetrics,
    LlmAsJudge,
)


# Define evaluation criterion with tool use metric
criterion = Criterion(
    name="tool_use_quality",
    description="Evaluates quality of tool selection and usage",
    metric=PrebuiltMetrics.RUBRIC_BASED_TOOL_USE_QUALITY_V1,
    threshold=0.7, # Require 70% tool use quality
)

# Create evaluator
evaluator = LlmAsJudge(
    model="gemini-2.0-flash", # LLM for evaluation
    criteria=[criterion],
)

# Evaluate agent execution
result = evaluator.evaluate(
    invocation=agent_result,
)
```


Parameter	Type	Purpose
<code>metric</code>	Enum	Set to <code>PrebuiltMetrics.RUBRIC_BASED_TOOL_USE_QUALITY_V1</code>
<code>threshold</code>	float	Min score (0-1) for pass/fail
<code>name</code>	str	Human-readable criterion name
<code>description</code>	str	What you're evaluating

Pro Tips

 **Tip 1 - Combine metrics:** Use tool use quality + final response quality for complete evaluation.

```
criteria = [
    Criterion(
        name="final_answer_quality",
        metric=PrebuiltMetrics.RUBRIC_BASED_FINAL_RESPONSE_QUALITY_V1,
        threshold=0.8,
    ),
    Criterion(
        name="tool_use_quality",
        metric=PrebuiltMetrics.RUBRIC_BASED_TOOL_USE_QUALITY_V1,
        threshold=0.7,
    ),
]
```

```
# Now you catch both:
# - Agents that get lucky with right answer, wrong method
# - Agents that use tools perfectly but get wrong answer
```

 **Tip 2 - Set realistic thresholds:** Tool use is complex; be generous with initial thresholds.

```
# Too strict - rarely passes
threshold=0.95 # ✗ Unrealistic

# Realistic baseline
threshold=0.7 # ✓ Allows some inefficiency

# Focus on improvements
threshold=0.8 # ✓ After you've optimized
```

 **Tip 3 - Log tool sequences for debugging:** When tool use quality is low, inspect the actual sequence.

```
# In your agent callback/logging
print("Tool sequence used:", [
    tool_call.name for tool_call in invocation.tool_calls
])

# Compare with optimal:
print("Optimal sequence:", expected_tools)

# Visualize differences to identify patterns
```

| When NOT to Use It

⚠️ Avoid when:

- Simple agents with only one tool (no sequencing to evaluate)
- Toolless workflows (e.g., pure reasoning agents)
- Tool usage is non-deterministic (truly no "right" order)
- Testing final answer correctness only (use Response Quality instead)

⚠️ Combine with:

- **Final Response Quality:** Check answer correctness independently
- **Safety Evaluator:** Ensure tool usage is compliant
- **Trajectory Evaluator:** Full execution path analysis

| Complete Working Implementation

The full implementation includes:

- Agent with multiple tools demonstrating different usage patterns
- Evaluation framework setup with criteria
- Comprehensive test suite validating evaluations
- Real example invocations to evaluate
- Development and testing utilities

```
cd til_implementation/til_rubric_based_tool_use_quality_20250121/  
make setup      # Install dependencies  
make test       # Run evaluation tests  
make evaluate   # ★ RUN REAL EVALUATION (NEW!)  
make dev        # Launch interactive evaluation UI  
make demo       # Quick validation
```

The `make evaluate` **Command (Real Evaluation):**

```
$ make evaluate
```

TEST CASES SUMMARY

Test Case 1: good_sequence_complete_pipeline

Description: Complete 4-step pipeline (analyze → extract → validate → apply)

Expected Score: 0.95-1.0 (excellent)

Test Case 2: bad_sequence_skipped_validation

Description: Missing steps (extract → apply, no analyze or validate)

Expected Score: 0.25-0.4 (poor)

Test Case 3: good_sequence_proper_analysis

Description: Good analysis pipeline (analyze → extract → validate)

Expected Score: 0.8-0.9 (good)

RUNNING EVALUATION

EVALUATION CONFIGURATION


Threshold: 0.7

Judge Model: gemini-2.5-flash

Rubrics: 4

- proper_tool_order: Agent calls analyze_data BEFORE extract_features...
- complete_pipeline: All 4 tools called in sequence...
- validation_before_model: Quality validated before modeling...
- no_tool_failures: All tool calls succeed with proper parameters...

[LLM Judge evaluates each test case against rubrics...]

 Evaluation ran but test cases failed scoring threshold:
Expected threshold: 0.7, actual value: 0.25

This means the evaluation framework is working correctly!
The test agent didn't match expected tool sequences.

In a real scenario, you would:

1. Review the expected vs actual tool calls above
2. Adjust agent instructions to match expected behavior
3. Re-run the evaluation to see if scores improve

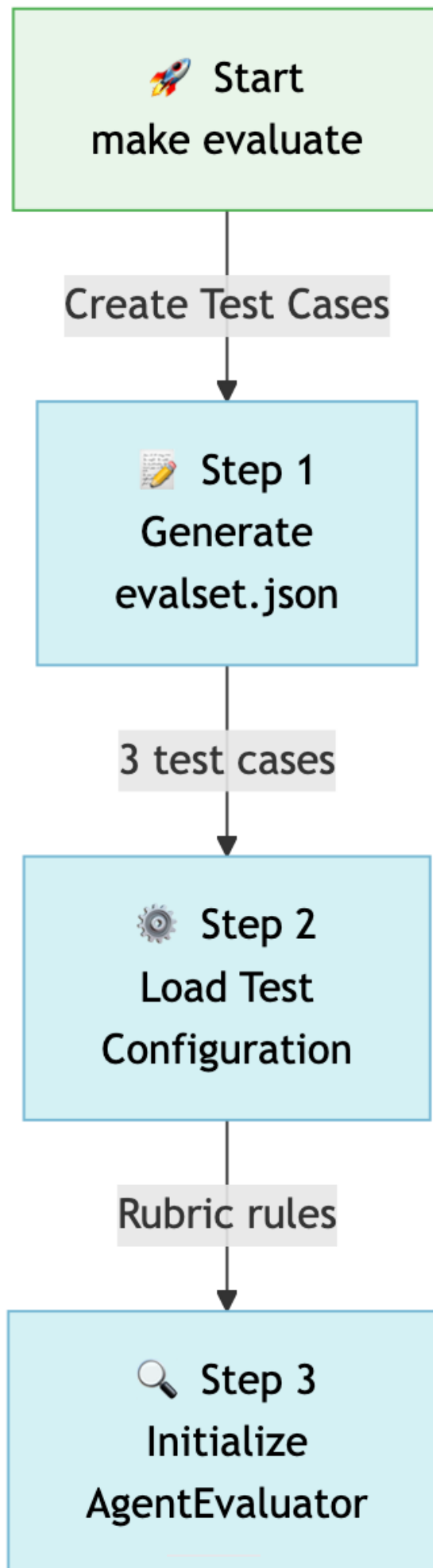
What's happening under the hood:

```
# evaluate_tool_use.py creates:
evalset = {
    "eval_cases": [
        {
            "eval_id": "good_sequence_complete_pipeline",
            "intermediate_data": {
                "tool_uses": [
                    {"name": "analyze_data", ...},
                    {"name": "extract_features", ...},
                    {"name": "validate_quality", ...},
                    {"name": "apply_model", ...}
                ]
            }
        },
        # ... bad and partial sequences
    ]
}

# Then calls:
results = await AgentEvaluator.evaluate(
    agent_module="tool_use_evaluator",
    eval_dataset_file_path_or_dir="tool_use_quality.evalset.json",
)

# Which internally:
# 1. Loads your agent
# 2. Compares actual tool calls vs expected
# 3. Uses Gemini as judge to evaluate against rubrics
# 4. Returns scores (0.0-1.0)
# 5. Shows pass/fail vs threshold
```

Complete evaluation workflow visualization:



Test the implementation:

```
# Run from implementation directory
pytest tests/ -v

# Expected output:
# test_agent.py::TestAgentConfiguration::test_agent_name PASSED
# test_agent.py::TestAgentConfiguration::test_agent_has_tools PASSED
# test_agent.py::TestToolFunctionality::test_analyze_data_success PASSED
# test_tool_use_evaluator.py::test_evaluation_framework PASSED
#
# 23 passed in 0.42s
```

| Real-World Example: Multi-Step Query

Here's how tool use quality reveals hidden problems:


```
# SCENARIO: Customer requests refund analysis

# AGENT EXECUTION:
def agent_workflow():
    # Step 1: Verify customer exists
    result1 = get_customer(customer_id="12345")

    # Step 2: Check order history
    result2 = get_orders(customer_id="12345")

    # Step 3: Calculate eligible refund
    result3 = calculate_refund(orders=result2.data)

    # Step 4: Process refund
    result4 = process_refund(customer_id="12345", amount=result3.amount)

    return result4

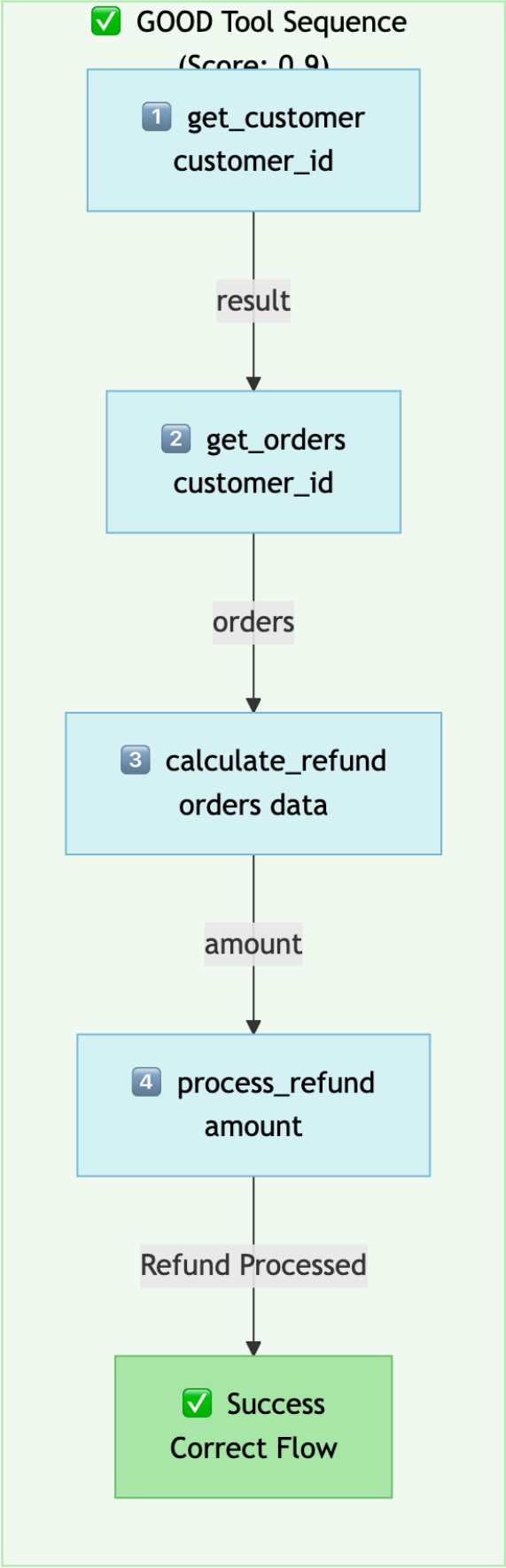
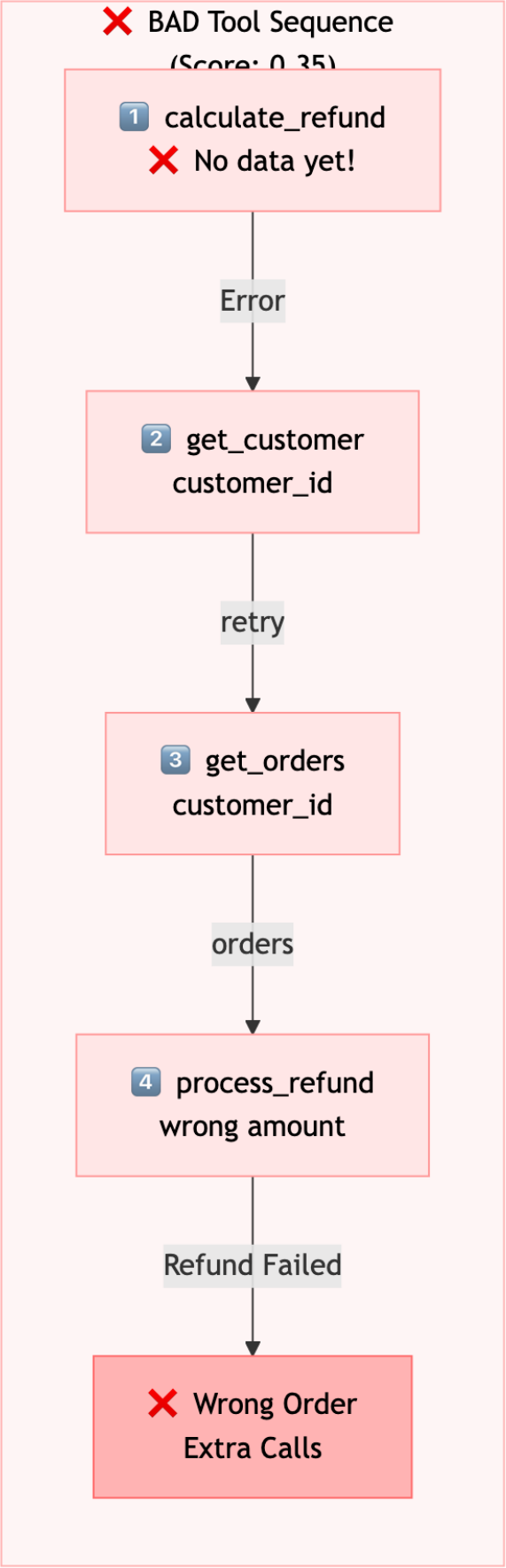
# EVALUATION RESULTS:

Final Response Quality: ✓ PASS (0.9)
- Correct refund amount
- Right customer ID
- Proper documentation

Tool Use Quality: ✗ FAIL (0.45)
- ✗ Called get_customer() but never used the result
- ✓ Correct sequence: Verify → Check Orders → Calculate → Process
- ✗ Could have combined verify + order check into single call
- ✗ get_customer() returned sensitive data not needed (security issue)

IMPROVEMENT NEEDED: Optimize tool usage, not answer correctness
```

Visual comparison: Good vs Bad tool sequencing:



Understanding Rubric Scoring

Tool Use Quality produces a **single score 0.0-1.0**:

- 1.0 = Perfect tool usage
 - Right tools for task
 - Optimal sequence
 - No waste or errors
 - Perfect recovery
- 0.8 = Very good
 - Right tools, minor inefficiency
 - One unnecessary call
 - Good error handling
- 0.5 = Adequate
 - Correct answer but poor tool choices
 - Some wrong tool uses
 - Backwards sequencing
- 0.0 = Failed
 - Wrong tools entirely
 - Tools called in wrong order
 - Unnecessary tool combinations

Next Steps After Learning

1. 📖 **Read related evaluation tutorial:** Tutorial on ADK evaluation framework for comprehensive understanding
2. 🚀 **Implement in your agent:** Use tool use quality criteria in your evaluation pipeline
3. 🛠️ **Combine with other metrics:** Build comprehensive evaluation combining multiple criteria
4. 📊 **Track metrics over time:** Monitor tool use quality as you improve agents

Key Takeaway

Tool Use Quality evaluation catches the hidden problems that final answer metrics miss.

Your agent might be getting the right answer through the wrong process. Tool Use Quality Metric forces you to care about HOW agents solve problems, not just IF they solve them.

Perfect for production agents where efficiency, compliance, and proper workflows matter as much as correctness.

Catch tool misuse before production. ✨

See Also

| Related TILs

- **[TIL: Pause and Resume Invocations](#)** ([til_pause_resume_20251020](#)) -
Use pause/resume to implement human-in-the-loop workflows that verify tool quality before proceeding. Perfect for evaluating agent decisions at checkpoints.
- **[TIL: Context Compaction](#)** ([til_context_compaction_20250119](#)) -
Ensure context compaction doesn't degrade tool selection quality. Monitor tool use quality metrics as context becomes more compressed.
- **[Back to TIL Index](#)** ([til_index](#)) - Browse all quick-learn guides

| Related ADK Tutorials

- **[Tutorial 01: Hello World Agent](#)** ([hello_world_agent](#)) -
Foundation for understanding agents that will be evaluated for tool quality
- **[Tutorial 02: Function Tools](#)** ([function_tools](#)) -
Learn how to design tools that agents will be evaluated on; crucial for tool use quality metrics
- **[Tutorial 06: Multi-Agent Systems](#)** ([multi_agent_systems](#)) -
Learn about combining agents efficiently; relevant to tool use quality in larger systems
- **[Tutorial 09: Evaluation & Testing](#)** ([evaluation_testing](#)) -
Comprehensive guide to all evaluation metrics and frameworks




ADK Official Documentation

- **Evaluation Framework** (<https://github.com/google/adk-python/blob/main/docs/evaluation.md>) -
Official ADK evaluation documentation
- **Tool Use Quality Metric** (<https://github.com/google/adk-python/blob/main/google/adk/evaluation/metrics.py>) -
Source code and implementation details
- **Agent Evaluator API** (https://github.com/google/adk-python/blob/main/google/adk/evaluation/agent_evaluator.py) -
Complete API reference for running evaluations

Related Resources & Patterns

- **Deploy AI Agents: Production Strategies** (</blog/deploy-ai-agents>) -
Understand quality assurance and evaluation in production systems
- **Building Reliable Agents: A Testing Strategy** (</blog/building-reliable-agents>) -
Comprehensive approach to agent quality including tool use evaluation

Questions?

-  Comment below with your evaluation questions
-  Found an issue? Check the implementation tests
-  Ready to evaluate your agent? See the implementation

Generated on 2025-10-21 09:03:44 from til_rubric_based_tool_use_quality_20251021.md

Source: Google ADK Training Hub