

Supplementary Materials: Agentic Test Suite Refinement

Rye Howard-Stone
rye.howard-stone@uconn.edu
University of Connecticut
USA

ACM Reference Format:

Rye Howard-Stone. 2025. Supplementary Materials: Agentic Test Suite Refinement. In *Proceedings of AI for Software Engineering (CSE 5095)*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 APPENDIX A: PROMPT VARIANTS

Note: The prompts shown below are *simplified summaries*. The actual prompts used in experiments were substantially larger (30–80 lines) and included pre-execution shell commands for context gathering. Full prompt text is available at:

<https://github.com/rhowardstone/aTSR/tree/main/src/prompts>

Table 1: Prompt Variant Summary

Variant	Lines	Description
V1: Baseline	30	Simple directive to improve coverage. No pre-execution context.
V2: +Context	40	Adds pre-execution commands: find, wc, ls to detect language, structure, test locations, code/test ratio.
V3: +Coverage	60	V2 + explicit coverage.py workflow with iterative gap-filling loop.
V4: +Mutations	60	V2 + explicit mutmut workflow for mutation-guided testing.
V5: +Both	80	V2 + three-phase workflow: coverage first, then mutations, then verification.
V6: Minimal	15	Pre-execution context only. Directive reduced to: "Improve coverage to 80%+."

Phase 1 (Base vs Refine): "Base" used a simple coverage improvement prompt; "Refine" used the full /refine-tests slash command implementing a 6-phase workflow (environment detection, coverage analysis, mutation testing, recommendations, property testing, verification). Full slash command source: <https://github.com/rhowardstone/aTSR/tree/main/claude/commands>

2 APPENDIX B: COMPLETE METRICS

2.1 Phase 1: Base vs Refine (12 configurations)

2.2 Phase 2: Six Prompt Variants (54 configurations)

3 APPENDIX C: STATISTICAL ANALYSIS CODE

```
import numpy as np
from scipy.stats import binomtest, wilcoxon, mannwhitneyu
```

```
# V4 underperformed V1 in 9/9 repos
```

CSE 5095, Fall 2025, University of Connecticut
2025. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

Table 2: Phase 1 Complete Results

Repo	Model	Strategy	Cov%	Pass%	Tests	Tokens
schedule	Sonnet	base	88	100.0	76	2.9M
schedule	Sonnet	refine	85	72.5	216	4.9M
schedule	Opus	base	91	100.0	94	3.2M
schedule	Opus	refine	90	96.8	212	6.0M
mistune	Sonnet	base	79	94.6	194	6.0M
mistune	Sonnet	refine	72	85.0	258	5.6M
mistune	Opus	base	76	94.3	376	4.2M
mistune	Opus	refine	71	97.7	300	7.9M
click	Sonnet	base	64	100.0	234	2.4M
click	Sonnet	refine	64	91.4	334	8.2M
click	Opus	base	67	91.9	212	10.6M*
click	Opus	refine	66	95.5	298	3.3M

* Anomaly: Extended exploration cycles integrating with 190 existing baseline tests.

Table 3: Phase 2 Experiment 1: Reduced Coverage Repositories

Repo	Variant	Cov%	Pass	Fail	Tokens (M)
click	V1	79	565	0	11.9
click	V2	66	340	2	5.1
click	V3	70	362	0	5.3
click	V4	0	0	0	5.5
click	V5	65	327	2	5.2
click	V6	63	341	0	5.9
mistune	V1	85	55	5	2.7
mistune	V2	64	28	0	4.3
mistune	V3	53	21	1	3.6
mistune	V4	46	18	2	4.4
mistune	V5	79	49	1	4.9
mistune	V6	82	52	0	4.9
schedule	V1	97	45	0	1.4
schedule	V2	94	38	0	2.5
schedule	V3	95	40	0	2.7
schedule	V4	56	22	3	4.0
schedule	V5	97	47	0	4.2
schedule	V6	88	35	0	0.9

```
v4_worse = 9
result = binomtest(v4_worse, 9, p=0.5, alternative='greater')
print(f"Sign test (V4 < V1): p = {result.pvalue:.4f}")
# Output: p = 0.0020
```

```
# Variance decomposition
total_var = 151.7
```

Table 4: Phase 2 Experiment 2: Low Existing Coverage Repositories

Repo	Variant	Cov%	Pass	Fail	Tokens (M)
python-box	V1	93	142	0	3.1
python-box	V2	94	89	0	3.1
python-box	V3	94	95	0	2.9
python-box	V4	1	2	0	3.6
python-box	V5	93	137	0	3.3
python-box	V6	95	148	0	3.7
colorama	V1	77	58	0	5.4
colorama	V2	80	42	0	5.0
colorama	V3	77	55	0	2.3
colorama	V4	67	38	0	3.4
colorama	V5	73	67	0	5.8
colorama	V6	72	71	0	7.3
boltons	V1	67	1205	12	6.4
boltons	V2	68	1198	8	3.4
boltons	V3	78	1245	5	7.7
boltons	V4	65	1180	15	4.0
boltons	V5	76	1232	3	7.6
boltons	V6	70	1210	7	7.4

Table 5: Phase 2 Experiment 3: Bioinformatics Repositories

Repo	Variant	Cov%	Pass	Fail	Tokens (M)
dnaapler	V1	35	12	8	5.5
dnaapler	V2	53	18	5	3.4
dnaapler	V3	34	11	9	7.7
dnaapler	V4	0	0	0	4.0
dnaapler	V5	38	14	6	5.5
dnaapler	V6	41	15	5	3.4
fastqe	V1	95	28	2	2.9
fastqe	V2	87	25	5	4.3
fastqe	V3	37	8	12	3.5
fastqe	V4	0	0	0	4.4
fastqe	V5	0	0	0	4.9
fastqe	V6	0	0	0	4.9
pyfaidx	V1	50	45	15	5.2
pyfaidx	V2	60	52	8	4.1
pyfaidx	V3	55	48	12	6.2
pyfaidx	V4	0	0	0	3.8
pyfaidx	V5	45	40	18	5.8
pyfaidx	V6	52	46	14	4.5

```

between_repo_var = 132.0
between_variant_var = 5.2
print(f"Ratio: {between_repo_var/between_variant_var:.1f}x")
# Output: 25.5x

# Token efficiency V5 vs V6
v5_tokens = [5.15, 4.89, 4.25, 3.27, 5.83, 7.62]
v6_tokens = [5.86, 4.92, 0.94, 3.72, 7.32, 7.38]
v6_wins = sum(v6 < v5 for v6, v5 in zip(v6_tokens, v5_tokens))
result = binomtest(v6_wins, 6, p=0.5, alternative='greater')
print(f"Sign test: p = {result.pvalue:.4f}")

```

```
# Output: p = 0.8906 (NOT significant)
```

```

# Domain effects
general = [79, 85, 97, 93, 77, 67]
bio = [35, 95, 50]
u_stat, u_p = mannwhitneyu(bio, general, alternative='less')
print(f"Mann-Whitney U: p = {u_p:.4f}")
# Output: p = 0.1905 (NOT significant)

```

4 APPENDIX D: TESTSCOUT EVALUATION RESULTS

Table 6: testScout Element Discovery Performance

Website	Type	Elements	Time (s)	Success
Wikipedia	Content Site	280	<5	✓
Hacker News	News Aggregator	226	<5	✓
GitHub Explore	Modern SPA	220	<5	✓
TodoMVC React	React App	12	<5	✓
Total		738		100%

Table 7: testScout Cost Comparison

Framework	Cost per Test	Notes
testScout (Gemini)	\$0.01–\$0.05	Pay-per-use
testScout (OpenAI)	\$0.10–\$0.50	Pay-per-use
Selenium/Playwright	\$0	Manual scripting
ZeroStep	\$49–\$249/mo	SaaS subscription
Checksum	\$99+/mo	SaaS subscription

5 APPENDIX E: FAILURE MODE AUTOPSIES

5.1 V4 Mutation Testing Trap (click)

Session log analysis reveals the agent’s descent into the mutation testing trap:

```

Tool 12: TodoWrite - "Install mutmut for mutation testing"
Tool 13: Bash - pip install mutmut
Tool 18: Bash - mutmut run --paths-to-mutate=src/click
ERROR: "No such option: --paths-to-mutate"
Tool 19-20: Bash - mutmut --help (reading docs)
Tool 22: Write - creates setup.cfg with mutmut config
Tool 23: Bash - timeout 300 mutmut run --simple-output
ERROR: "No such option: --simple-output"
Tool 25-26: Bash - mutmut results / ls .mutmut*
OUTPUT: "No mutmut files found"

```

When mutmut finally ran, it generated hundreds of unchecked mutants:

```

click._compat.x__make_text_stream__mutmut_1: not checked
click._compat.x__make_text_stream__mutmut_2: not checked
... (hundreds more)

```

Agent’s realization (message 69): “Let me try a different approach... The mutation run didn’t complete properly.”

Root Causes: (1) mutmut 3.x changed CLI options; agent knowledge was outdated. (2) No early exit strategy when tooling failed.

(3) Mutant explosion overwhelmed testing process. (4) 10+ tool calls on setup before any test writing.

5.2 10.6M Token Anomaly (click/Opus/base)

Token breakdown reveals why Opus used 4.4× more tokens than Sonnet:

Metric	Sonnet	Opus
Messages	47	161
Total tokens	2.4M	10.6M
Edit calls	12	25
Grep calls	0	12

Activity log shows repeated fix cycles:

Msg 40: "The base class expects source_template..."
 Msg 70: "Let me fix these issues:"
 Msg 80: "Let me debug this test..."
 Msg 120: "Let me fix the LazyFile test issue:"

Root Cause: Click had 190 existing tests. Opus’s thoroughness led to more exploration (12 Grep vs 0), more iterations (25 Edit vs 12), and compounding context costs.

Lesson: Sonnet achieved similar coverage (64% vs 67%) with 4.4× fewer tokens.

5.3 V4 Catastrophic Failures (All Repos)

In 5/9 repositories, V4 achieved $\leq 10\%$ coverage:

- **click:** 0% (mutation testing never completed)
- **python-box:** 1% (mutmut errors on complex types)
- **dnaapler:** 0% (bioinformatics domain confusion)
- **fastqe:** 0% (emoji handling in mutants)
- **pyfaidx:** 0% (file format parsing mutants)

Common pattern: Agent enters mutation analysis loops, generating hundreds of mutants, then exhausts context attempting to address each before writing basic coverage tests.

5.4 Test Quality Comparison: V1 vs V6 (schedule)

Both variants produced tests with similar quality characteristics:

Metric	V1 (Baseline)	V6 (Minimal)
Test file size	856 lines	610 lines
Test functions	~40	~30
Coverage	97%	88%
Tokens used	4.9M	6.4M

V1 Example (tests edge cases with real functions):

```
def test_job_repr_with_args_and_kwargs(self):
    with mock_datetime(2014, 6, 28, 12, 0):
        def my_job(arg1, arg2, kwarg1=None):
            pass
        job = every(1).hour.do(my_job, "test_arg", 42,
                               kwarg1="test_kwarg")
        assert "my_job" in repr(job)
        assert "'test_arg'" in repr(job)
```

V6 Example (simpler, uses mocks):

```
def test_job_repr_representation(self):
```

```
    with mock_datetime(2014, 6, 28, 12, 0):
        mock_job = make_mock_job(name="test_job")
        job = every(1).second.do(mock_job)
        assert "Every 1 second do test_job()" in repr(job)
```

Observation: V1 tests more edge cases but both rely heavily on `mock.Mock()`. Neither variant produces the ideal “real execution” tests that would provide higher confidence. This is a limitation of LLM-generated tests that future work should address.

6 APPENDIX F: JOBSOACH BUG REPORTS

Bugs detected by testScout during JobsCoach development:

- (1) **P0 - Route Ordering:** `/v2/jobs/fresh` caught by `/v2/jobs/{job_id}` dynamic route
- (2) **P1 - Database Init:** `create_all()` without `checkfirst=True` causing table exists errors
- (3) **P1 - Test Isolation:** 101 tests failing in suite but passing individually
- (4) **P1 - API Parameters:** query and filters silently ignored
- (5) **P2 - Duplicate Text:** “Senior Senior Software Engineer” in headlines
- (6) **P2 - Filter State:** Frontend/backend property mismatch

All bugs were identified through systematic E2E testing and subsequently fixed.

7 APPENDIX G: REPRODUCIBILITY

All experimental data, prompts, and analysis scripts are available at:

- **aTSR:** <https://github.com/rhowardstone/aTSR>
 - `results/phase1/summary.json` - Phase 1 metrics
 - `results/phase2/*/metrics.json` - Phase 2 metrics
- **testScout:** <https://github.com/rhowardstone/testscout>

Model versions used:

- Claude Sonnet 4.5 (claude-sonnet-4-5-20250901) – aTSR experiments
- Claude Opus 4.1 (claude-opus-4-1-20250901) – aTSR experiments
- Gemini 2.5 Pro – testScout AI mode experiments