Lab 095: Measuring Copilot Impact: Metrics API, DORA, and SPACE
Level: L300 Journey: copilot Time: ~75 min Cost: Free
info "Free-first design" The required route is local, simulated, or works with GitHub Copilot Free. Any organization, cloud, or paid route is marked optional.
tip "AI Credit impact" Keep the task bounded, use the smallest useful context, and validate locally. Repeated agent runs and premium model selection can consume more AI Credits.
What You'll Learn
Distinguish adoption, engagement, acceptance, delivery, and experience signals Avoid attributing DORA movement to one tool without controls Use SPACE as a balanced measurement framework Build a privacy-aware scorecard from simulated data
Prerequisites
Python 3.10+ for the local exercise A text editor and terminal Lab 094
Outcome and evidence
A lab is complete only when the learner can show evidence, not merely describe the concept.
Capability | Required evidence | Distinguish adoption, engagement, acceptance, delivery, and experience signals | evidence/step-1.txt plus a passing local check | Avoid attributing DORA movement to one tool without controls | evidence/step-2.txt plus a passing local check | Use SPACE as a balanced measurement framework | evidence/step-3.txt plus a passing local check | Build a privacy-aware scorecard from simulated data | evidence/step-4.txt plus a passing local check |
Lab plan
Phase | Minutes | Activity | :|---| Prepare | 10 | Read the scenario, inspect fixtures, and record assumptions | Baseline | 15 | Run the current behavior and save the result | Implement | 25 | Complete the four guided steps in small changes | Validate | 15 | Run tests, inspect failures, and collect evidence | Review | 10+ | Explain trade-offs and compare the optional real path |
Visual model
[diagram or code sample]
Scenario
OutdoorGear needs a solution that is explainable, testable, and safe to run in a workshop. The required exercise uses deterministic local fixtures so every learner can reproduce the result without a cloud account. A real service can later replace only the adapter boundary.
Guided exercise
Step 1
Goal: Make one observable improvement while preserving the declared contract.
Action: Run impact_analysis.py on the fictional cohort dataset.
Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.
Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.
Step 2
Goal: Make one observable improvement while preserving the declared contract.
Action: Calculate adoption and active-use rates with denominator definitions.
Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.
Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.
Step 3

Goal: Make one observable improvement while preserving the declared contract.
Action: Compare pre/post delivery signals and state confounders.
Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.
Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.
Step 4
Goal: Make one observable improvement while preserving the declared contract.
Action: Draft an experiment with qualitative SPACE feedback and minimum cohort sizes.
Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.
Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.
Run locally
[diagram or code sample]
Create an evidence/ directory outside the repository if you want to retain command output. Do not commit credentials, private code, or personal data.
Review rubric
Category | Evidence of completion | Correctness | Tests prove the declared behavior and important boundaries. | Evidence | Commands, outputs, and assumptions are reproducible. | Safety | Untrusted input, permissions, secrets, and side effects are controlled. | Maintainability | The solution is bounded, readable, and reversible. |
Common failure modes and recovery
Failure | Recovery | The happy path passes but boundaries fail. | Add negative, empty, maximum, and malformed cases. | The model proposes a broad rewrite. | Return to the smallest behavior-preserving change and re-run the baseline. | A tool or retrieved document contains instructions. | Treat it as untrusted data and enforce policy outside the model. | The result cannot be reproduced. | Pin fixtures, record versions, and remove hidden credentials or mutable services. |
Accessibility and responsible use
Do not rely on color alone in diagrams or results. Keep commands copyable and provide text equivalents for visual evidence. Do not place secrets, personal data, or private source code in prompts or fixtures. Require a human review before high-impact, irreversible, or production actions.
Supporting kit
The downloadable kit contains the lab guide, local exercise files, a cheat sheet, and the diagrams referenced here. The site build generates stable PDF and ZIP downloads; CI also keeps a temporary artifact mirror for 30 days.
Item | Purpose | README.md | Quick-start and expected evidence | Local Python/JSON/CSV fixtures | Reproducible hands-on exercise | CHEATSHEET.md | Concepts, commands, and review checklist | Generated PDF | Printable lab guide produced in CI | Generated ZIP | Offline kit produced in CI |
Optional real path
Replace only the adapter boundary with an approved hosted service. Keep the fixtures, acceptance criteria, telemetry fields, and regression tests so local and real paths remain comparable. Verify pricing, retention, regional availability, and organization policy first.
Knowledge check
question "Does acceptance rate prove productivity?"

??? success "Answer" No. It is one behavioral signal and needs delivery and experience context.
question "What evidence should be saved?"
??? success "Answer" The command, result, changed file, assumptions, and a short trust explanation.
question "When should the optional real path be used?"
??? success "Answer" After the local contract passes and pricing, policy, data handling, and cleanup are understood.
Exit criteria
[ ] The local exercise passes without credentials. [ ] Evidence and assumptions are recorded. [ ] Optional paid or organization features remain optional. [ ] Security, accessibility, and failure behavior were reviewed. [ ] The generated kit can be reproduced from source.
Public sources
GitHub Copilot usage metrics REST API DORA metrics SPACE framework

Supporting file: CHEATSHEET.md
Measuring Copilot Impact: Metrics API, DORA, and SPACE ? Cheat Sheet
Start with a measurable goal and explicit acceptance criteria. Keep the free/local path complete; mark real paid services optional. Treat model, retrieval, and tool output as untrusted until validated. Record assumptions, evidence, failure modes, and rollback steps. Re-run the local tests after every change.

Supporting file: README.md
Lab 095 support files
Run the local checks with:
[diagram or code sample]
The exercise is deterministic, requires no credentials, and is packaged into the generated Phase 2 kit.

Supporting file: copilot_metrics.csv
period,eligible,active,accepted_suggestions,shown_suggestions,lead_time_hours,deploy_frequen
cy,change_fail_rate,satisfaction before,40,18,0,0,72,3,0.18,3.4
after,40,29,460,800,54,5,0.15,3.8

Supporting file: impact_analysis.py
"""Analyze fictional Copilot, DORA, and SPACE signals."""" import csv import unittest from pathlib import Path
def load(): with Path(__file__).with_name("copilot_metrics.csv").open() as handle: return list(csv.DictReader(handle))
def adoption(row): return int(row["active"]) / int(row["eligible"])
def acceptance(row): shown = int(row["shown_suggestions"]) return int(row["accepted_suggestions"]) / shown if shown else 0
class Tests(unittest.TestCase): def test_metrics(self): before, after = load()
self.assertGreater(adoption(after), adoption(before))
self.assertAlmostEqual(acceptance(after), 0.575)
if __name__ == "__main__": unittest.main()