Unified Quality Feedback Across CI/CD Pipelines

From Jenkins plugins to GitHub Actions, GitLab CI, and Autograding

Prof. Dr. Ullrich Hafner
University of Applied Sciences Munich

HM

From Industry to Education

Prof. Dr. Ullrich Hafner
ullrich.hafner@hm.edu
Department of Computer Science
Professor of Software Engineering

- Jenkins plugin developer for ~20 years
  - Warnings
  - Coverage
  - Git Forensics
- Quality feedback for industrial CI pipelines
- Later: teaching at the university
  - software engineering
  - software development

CI Builds Software — Quality Feedback Is Fragmented

- Many tools produce similar quality data
- Multiple reports per build and per tool
- No consistent interpretation across builds

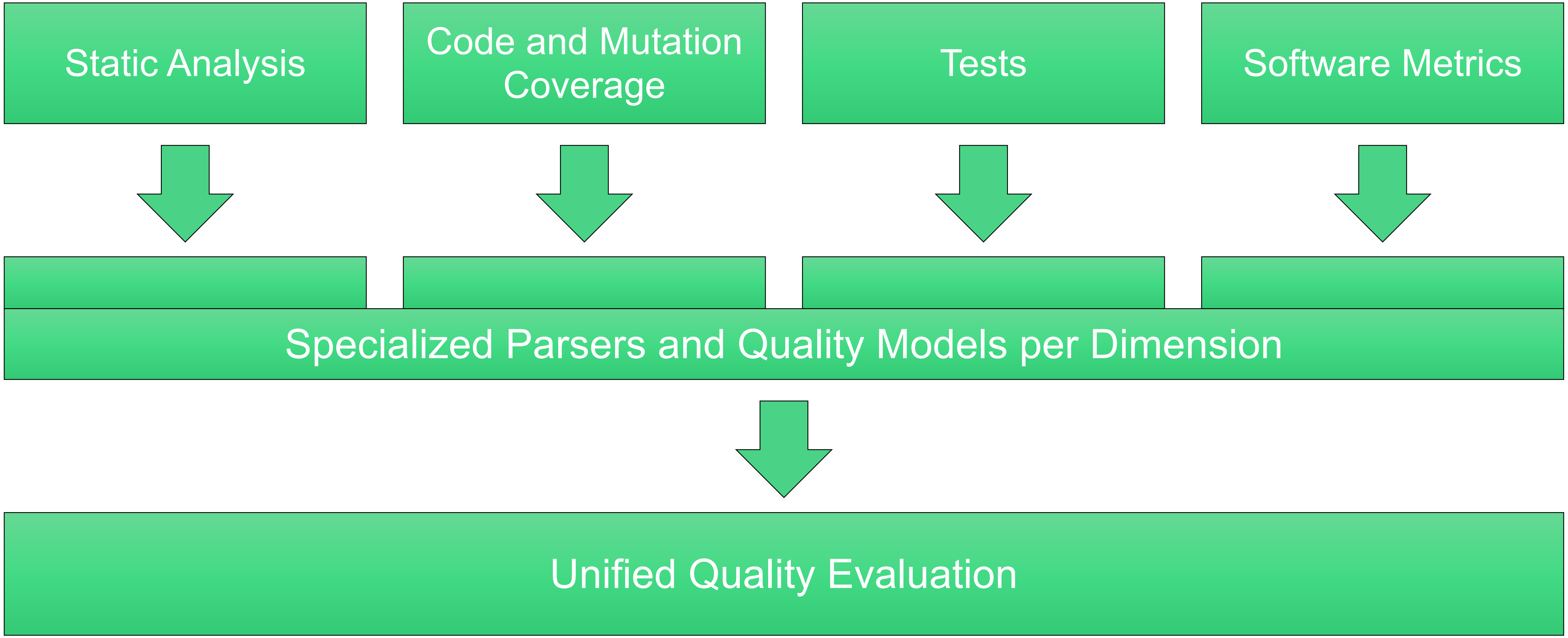
Jenkins Exposed the Need for a Shared Quality Model

- Many plugins for similar quality data
- Overlapping responsibilities across plugins
- The missing piece: a shared quality model across builds
- Enables aggregation, trends, and quality gates

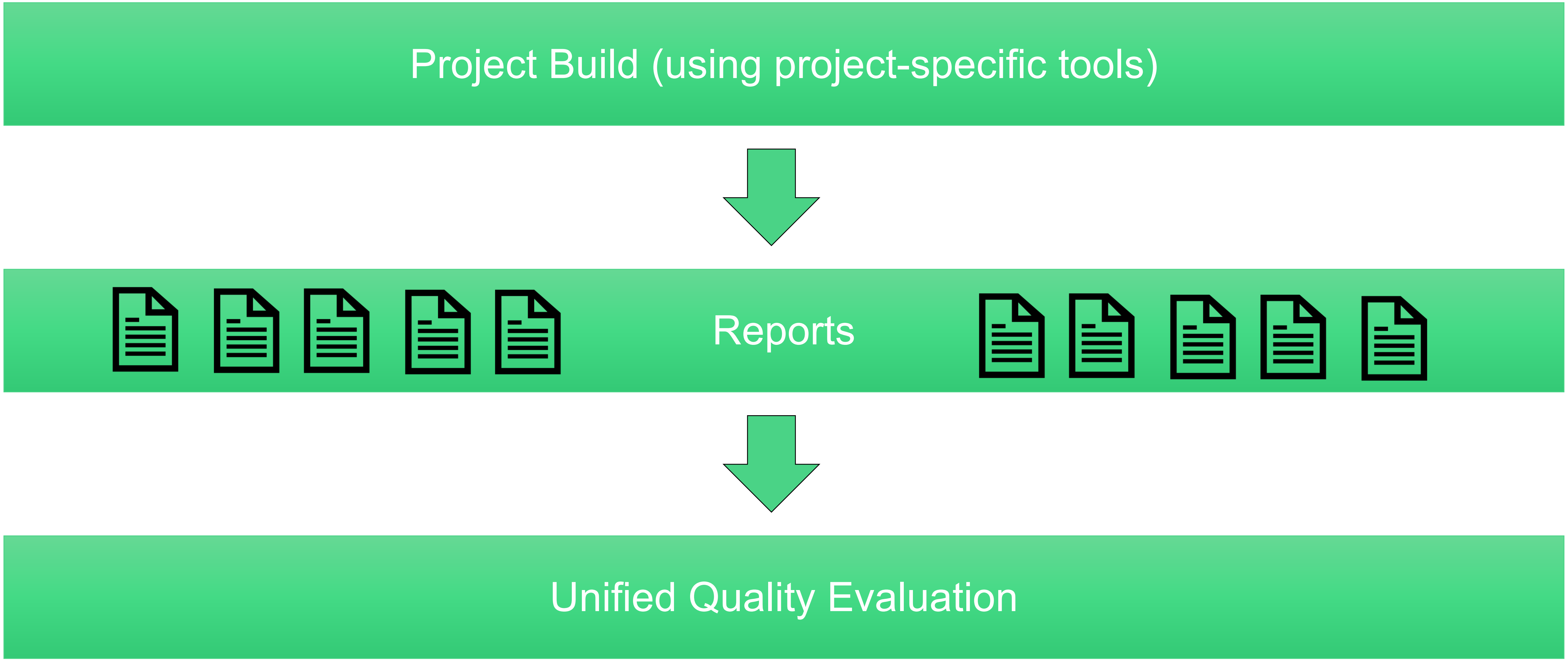
A Shared Quality Model Enables Multi-CI Feature Parity

- Jenkins remains the reference implementation
- The same quality model is reused across CI systems
- Jenkins, GitHub, and GitLab provide the same features
- Unified semantics across all CI frontends

Quality Dimensions Are Evaluated Separately — On Purpose


Static Analysis
Code and Mutation Coverage
Tests
Software Metrics
Specialized Parsers and Quality Models per Dimension
Unified Quality Evaluation

Quality Evaluation Is Decoupled from the Build


Project Build (using project-specific tools)
Reports
Unified Quality Evaluation

Unified Quality Feedback Across CI Systems

Unified Quality Evaluation

↓

Scores | Quality Gates | Trends

↓

Consistent Feedback (same content, same semantics)

↓

Jenkins UI | Pull Request Comments | Markdown

Flexible Quality Gates Enable Continuous Improvement

- Quality Gates can be defined as:
  - Absolute thresholds

    e.g. ≥ 70% global line coverage
  - Relative thresholds

    e.g. ≥ 80% of changed code
  - Delta-based thresholds

    e.g. -2% compared to main branch

Pull Requests Are the Right Place for Quality Feedback

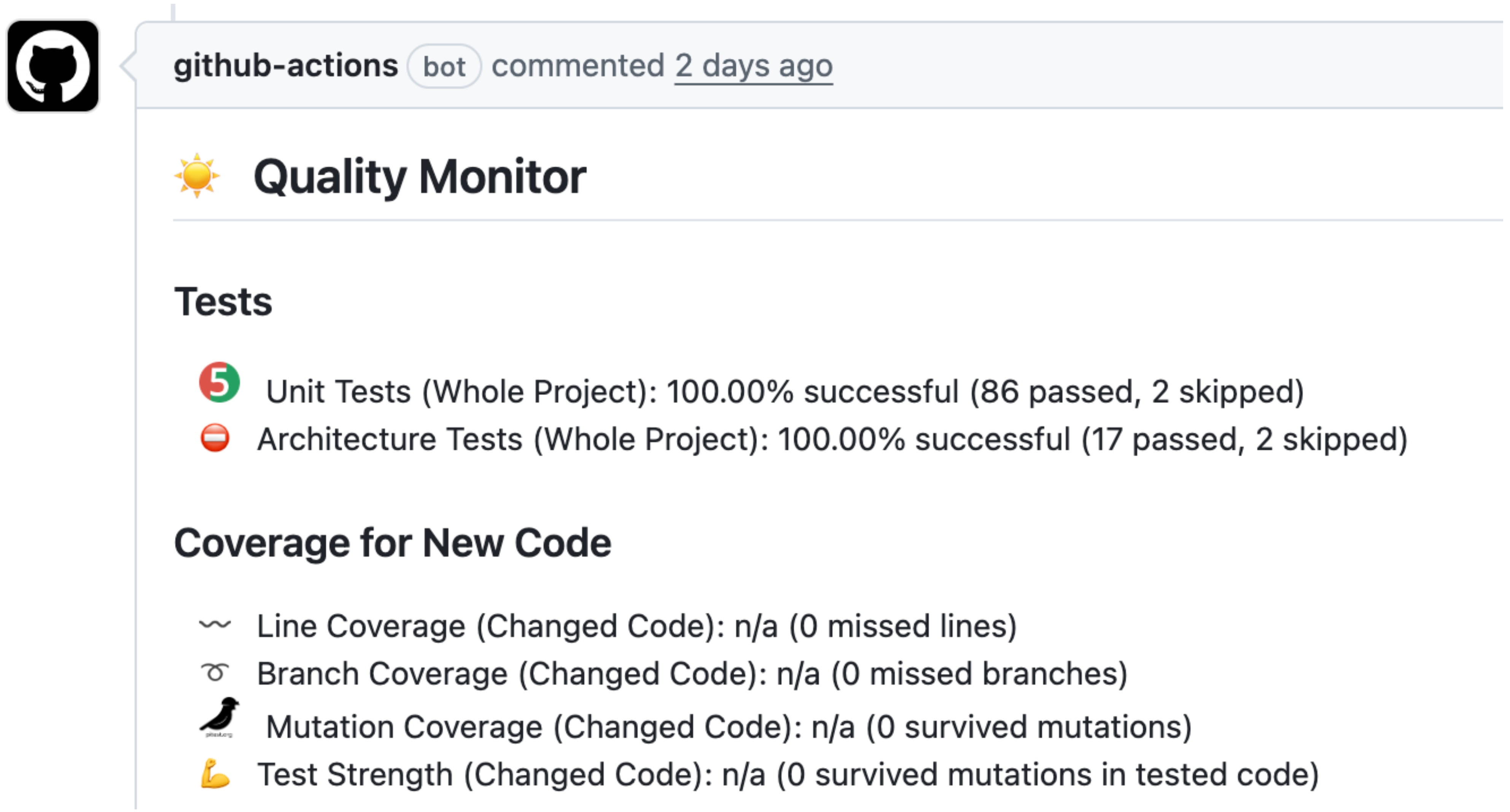
Unified Quality Feedback (same content, same semantics)

$\Downarrow$

Structured Pull Request Feedback

- Scores, gates, and trends per change
- Concise Markdown summaries
- Context for review decisions

Summary Feedback as a Pull Request Comment



github-actions bot commented 2 days ago

☀️ Quality Monitor

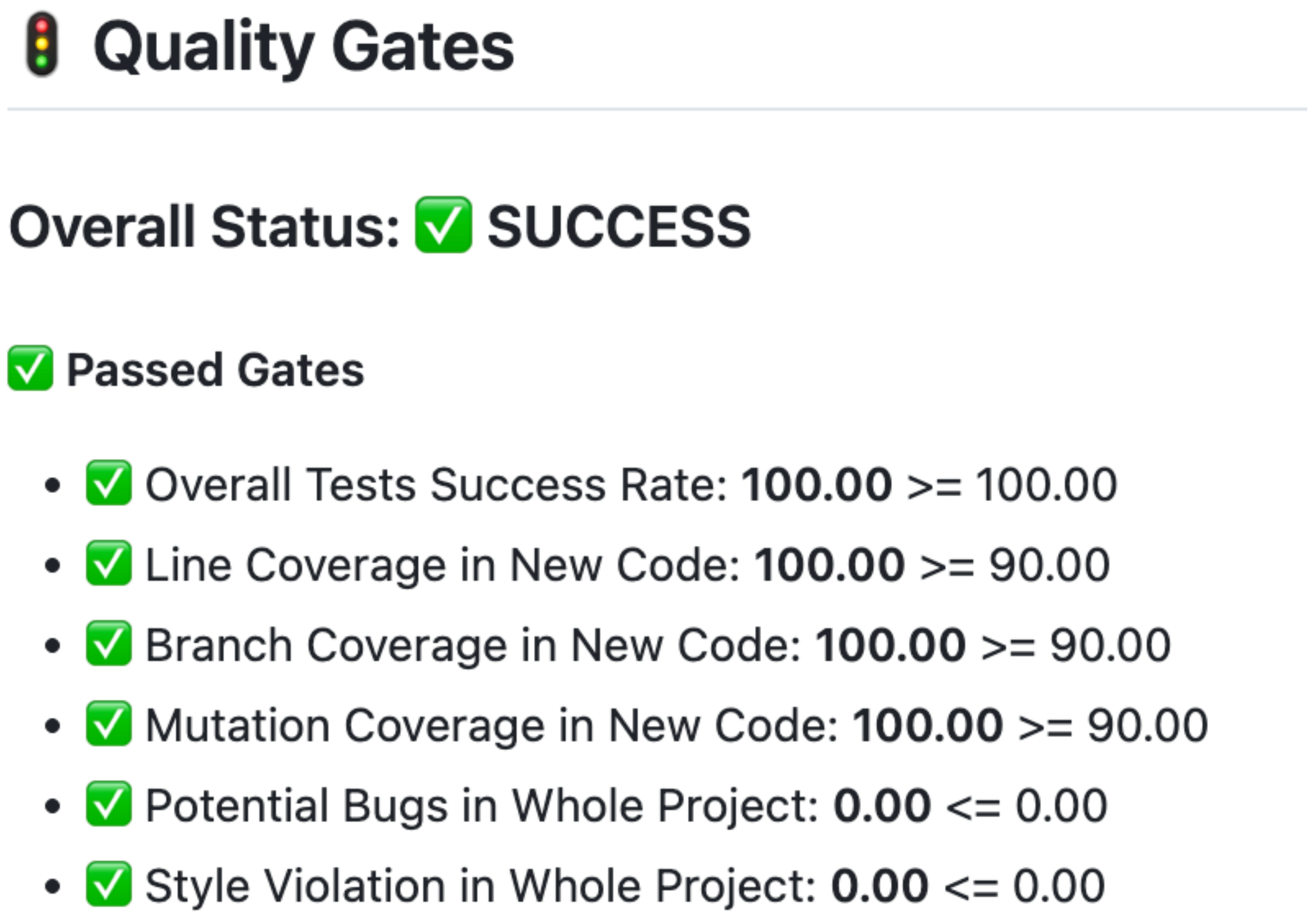
Tests

- Unit Tests (Whole Project): 100.00% successful (86 passed, 2 skipped)
- ⛔ Architecture Tests (Whole Project): 100.00% successful (17 passed, 2 skipped)

Coverage for New Code

- 〰️ Line Coverage (Changed Code): n/a (0 missed lines)
- ➰ Branch Coverage (Changed Code): n/a (0 missed branches)
- Mutation Coverage (Changed Code): n/a (0 survived mutations)
- 💪 Test Strength (Changed Code): n/a (0 survived mutations in tested code)

Quality Gate Result as a Pull Request Comment



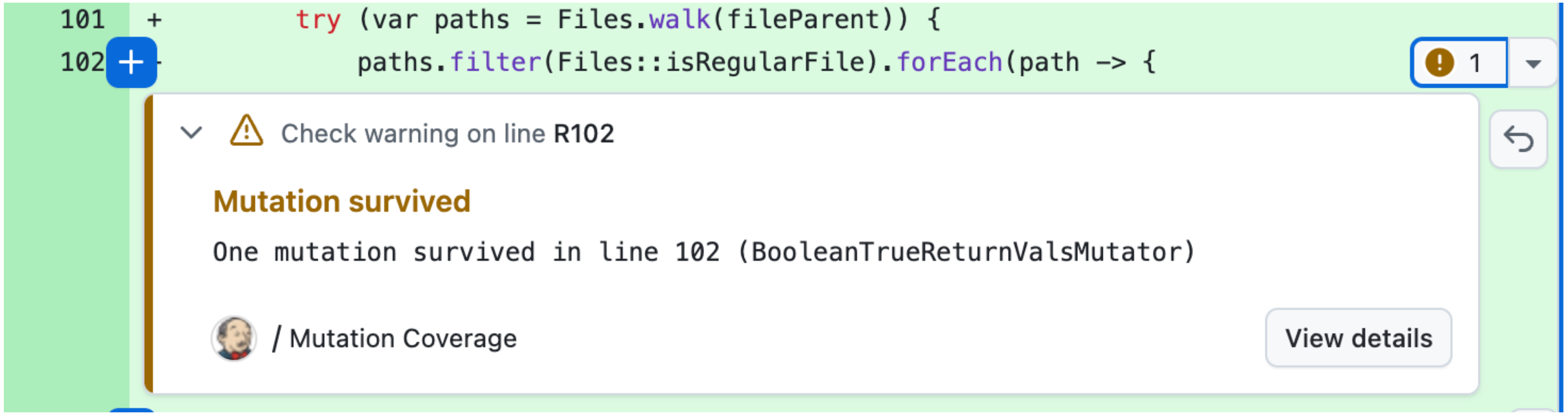
🚦 Quality Gates

Overall Status: ✅ SUCCESS

✅ Passed Gates

- ✅ Overall Tests Success Rate: **100.00** >= 100.00
- ✅ Line Coverage in New Code: **100.00** >= 90.00
- ✅ Branch Coverage in New Code: **100.00** >= 90.00
- ✅ Mutation Coverage in New Code: **100.00** >= 90.00
- ✅ Potential Bugs in Whole Project: **0.00** <= 0.00
- ✅ Style Violation in Whole Project: **0.00** <= 0.00

Detail Feedback as a Diff Annotation



```
+        try (var paths = Files.walk(fileParent)) {
+            paths.filter(Files::isRegularFile).forEach(path -> {
```

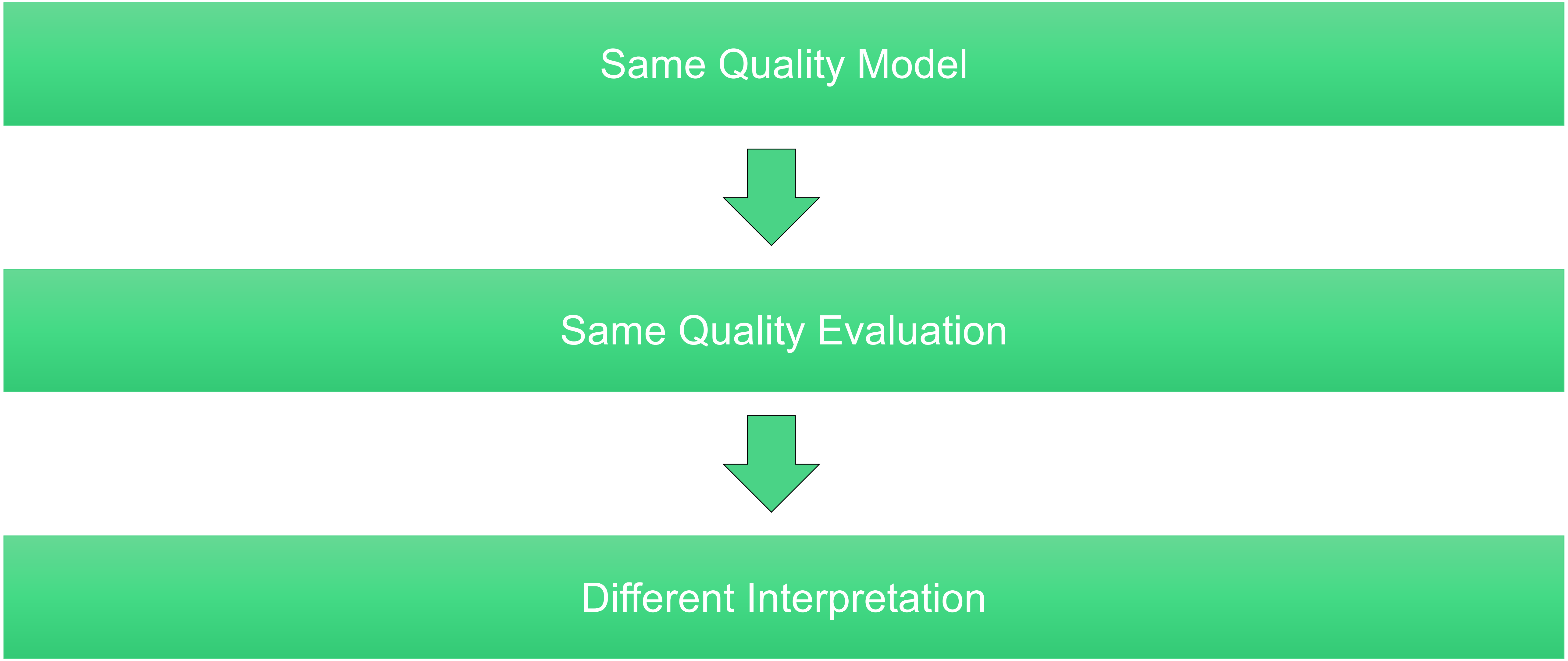
Check warning on line **R102**

Mutation survived

```
One mutation survived in line 102 (BooleanTrueReturnValsMutator)
```

/ Mutation Coverage

Autograding Is Quality Evaluation — Reinterpreted



Industry and Education Share the Same Quality Model

Industry CI	Education CI
Quality gates	Points
Release decisions	Grades
Quality trends	Learning progress
Pull request review	Student feedback

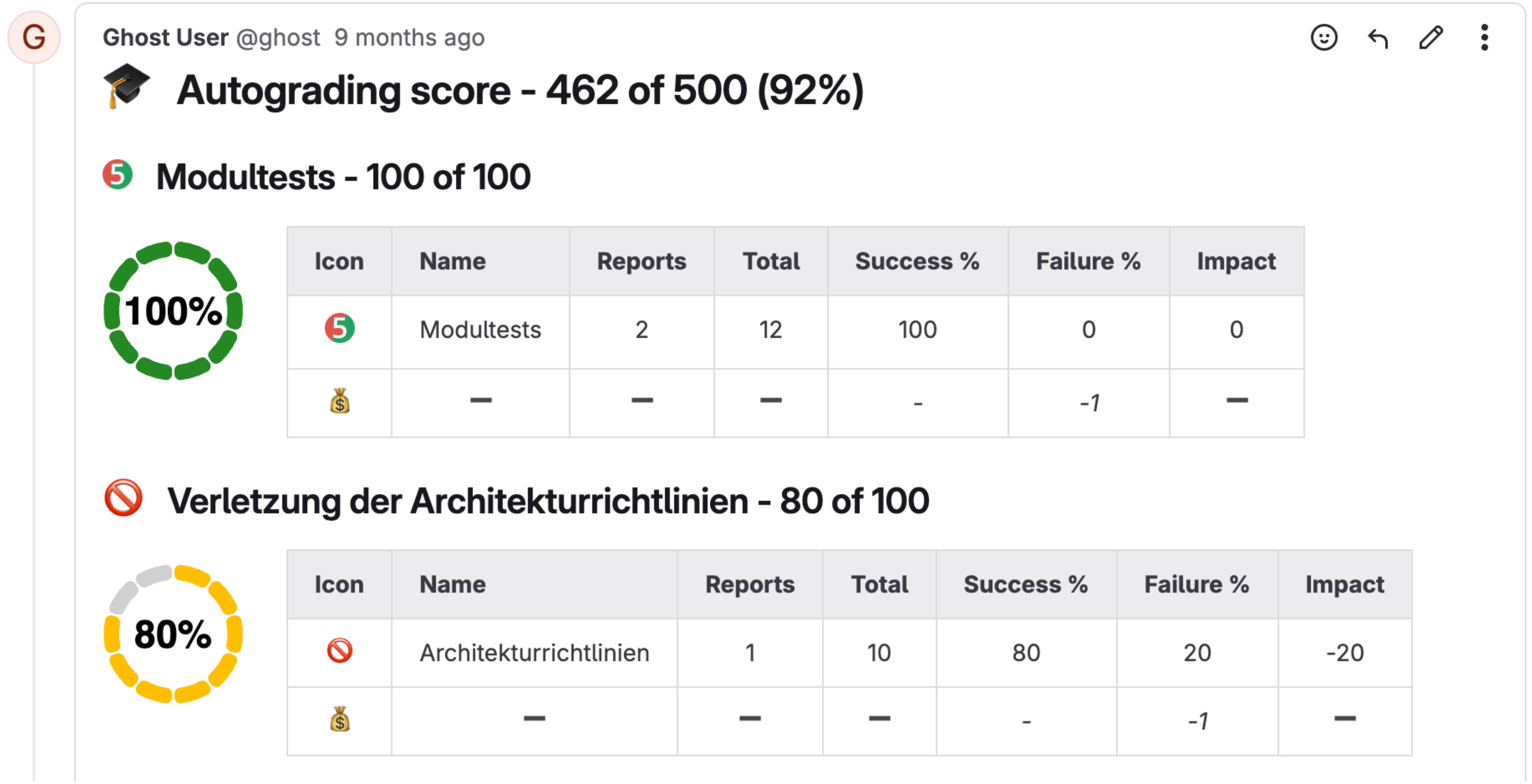
Lessons Learned from Applying One Model Across CI Systems

- CI systems differ less than expected
- Quality models outlive UIs
- Normalization enables portability
- Scoring is more flexible than pass/fail
- Shared cores reduce long-term maintenance

Quality Feedback Should Be Portable

- Independent of CI systems
- Independent of UIs
- Open source
- **CI pipelines build software — quality feedback should travel with it.**
  - https://github.com/uhafner/quality-monitor
  - https://github.com/uhafner/autograding-gitlab-action
  - Contributions are welcome!

Backup – Autograding Results in GitLab



Ghost User @ghost 9 months ago

🎓 Autograding score - 462 of 500 (92%)

5️⃣ Modultests - 100 of 100

Icon	Name	Reports	Total	Success %	Failure %	Impact
5️⃣	Modultests	2	12	100	0	0
💰	—	—	—	-	-1	—

🚫 Verletzung der Architekturrichtlinien - 80 of 100

Icon	Name	Reports	Total	Success %	Failure %	Impact
🚫	Architekturrichtlinien	1	10	80	20	-20
💰	—	—	—	-	-1	—