Selenium WebDriver

Hanieh Salmantaheri, Tania Sanjid, Numan Shaikh, Heeba Shaikh

Team I

What is Selenium WebDriver?

Open-source, W3C-compliant browser-automation **bridge** that turns **test code** into **real user actions** (click, type, navigate) across Chrome, Firefox, Edge, Safari, and more. It supports multiple languages, plugs smoothly into CI/CD pipelines, and scales from a local laptop to full cloud grids for enterprise-grade testing.

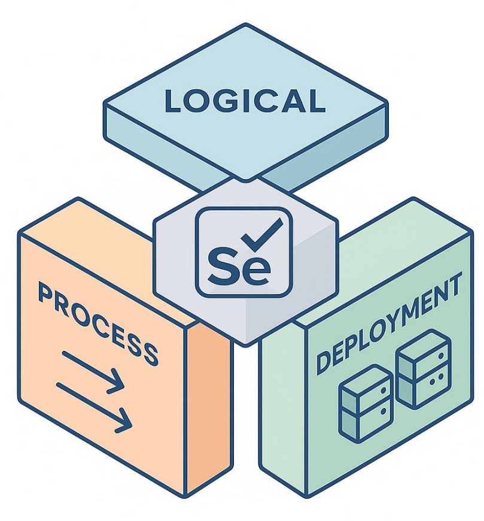


Figure 1: Three Orthogonal Views (ISO 42010)

Why Architects Care?

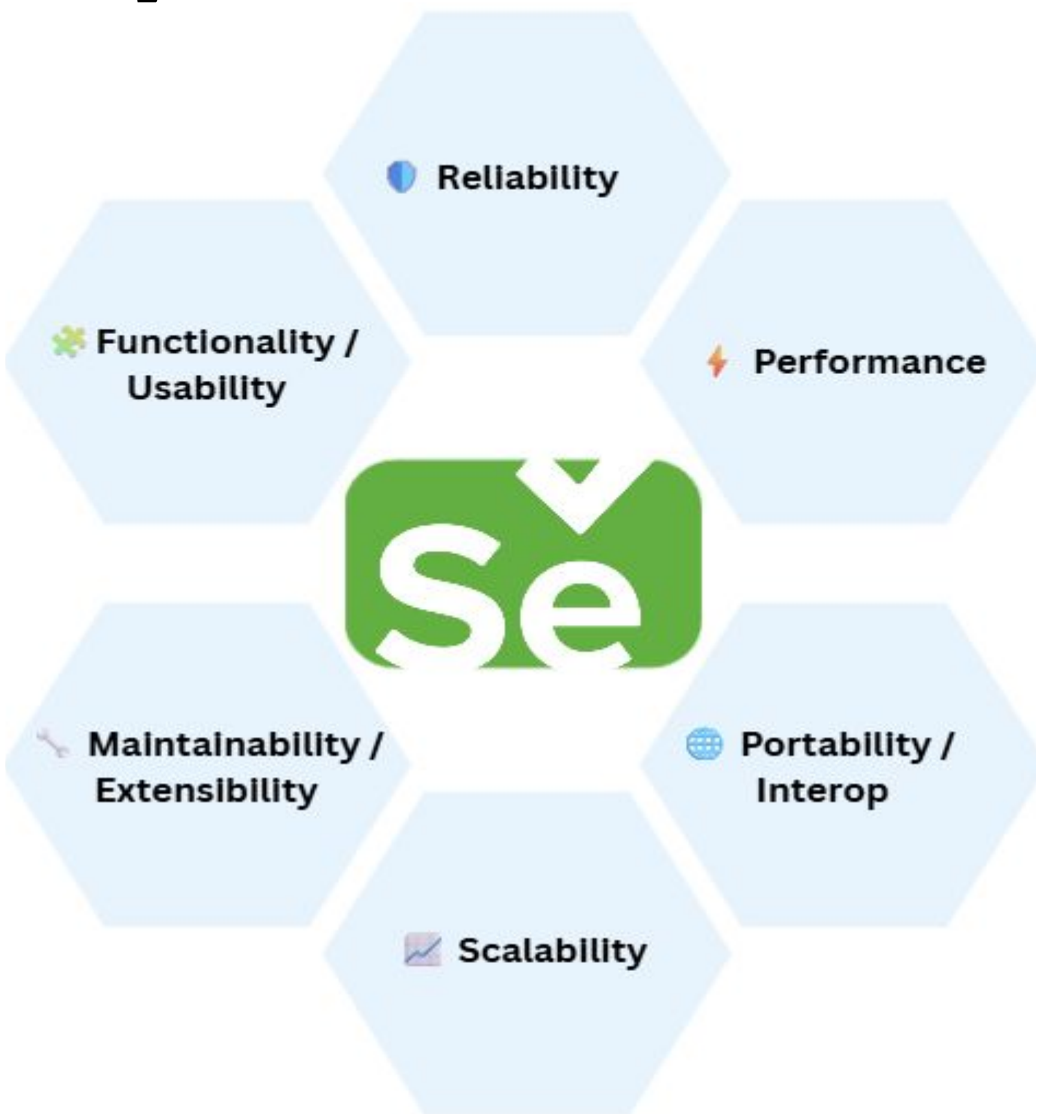


Figure 2: Quality Attributes

Selenium WebDriver hits the major ISO 25010 marks: it delivers full-stack **functionality** (click, type, navigate via a W3C API) that's **usable** even for beginners thanks to clean syntax and huge community docs. Tests stay **reliable** through isolated sessions and clear exceptions, while lean HTTP calls keep **performance** steady. Scripts are effortlessly **portable / interoperable** across OSs, languages, CI/CD tools, and cloud grids, and the open-source, modular core guarantees **maintainability + extensibility**. Built-in logs/screenshots aid **testability**; Grid and Docker provide real **scalability**; HTTPS-secured drivers offer basic **security**; and strict W3C compliance ensures lasting **compatibility**.

Selenium WebDriver Architecture

Architectural Patterns

C Client-Server Pattern
WebDriver sends HTTP commands to browser drivers for remote execution

L Layered Architecture
Separates API, language bindings, drivers, and browser layers

Architectural Principles

S Separation of Concerns
Each component handles specific responsibilities (bindings, execution, DOM)

O Open/Closed Principle
Components can be extended without modifying existing stable code

Architectural Styles

R RESTful Style
Uses JSON-over-HTTP protocol for stateless communication

C Component-Based Style
Interfaces like TakesScreenshot, HasCapabilities enable reusability

Architectural Tactics

A Asynchronous
Parallel testing execution

I Interface Abstraction
Component swapping

ANTI PATTERNS

⚠ HIGH IMPACT

Layered Monolithic Core

ARCHITECTURAL PAIN
All layers version-locked, slow round-trips, flaky tests

FIX
Shift to Selenium 4 W3C protocol or hide driver behind a façade

HIGH IMPACT

God-class Page Objects

ARCHITECTURAL PAIN
Low cohesion; small UI tweak → many edits (shotgun surgery)

FIX
Split into Page Fragments or Screenplay tasks

MEDIUM IMPACT

Duplicated Language Bindings

ARCHITECTURAL PAIN
Bug fixed in one binding, lingers in others

FIX
Generate bindings from one spec or expose shared logic as service

HIGH IMPACT

Selector Shotgun
(fragile locators)

ARCHITECTURAL PAIN
DOM re-layout breaks dozens of tests

FIX
Enforce data-test IDs & central Element Repository

MEDIUM IMPACT

Hard Waits / Thread.sleep()

ARCHITECTURAL PAIN
Bloats runtime; still races

FIX
Replace with explicit/fluent waits; wrap in Synchroniser util

HIGH IMPACT

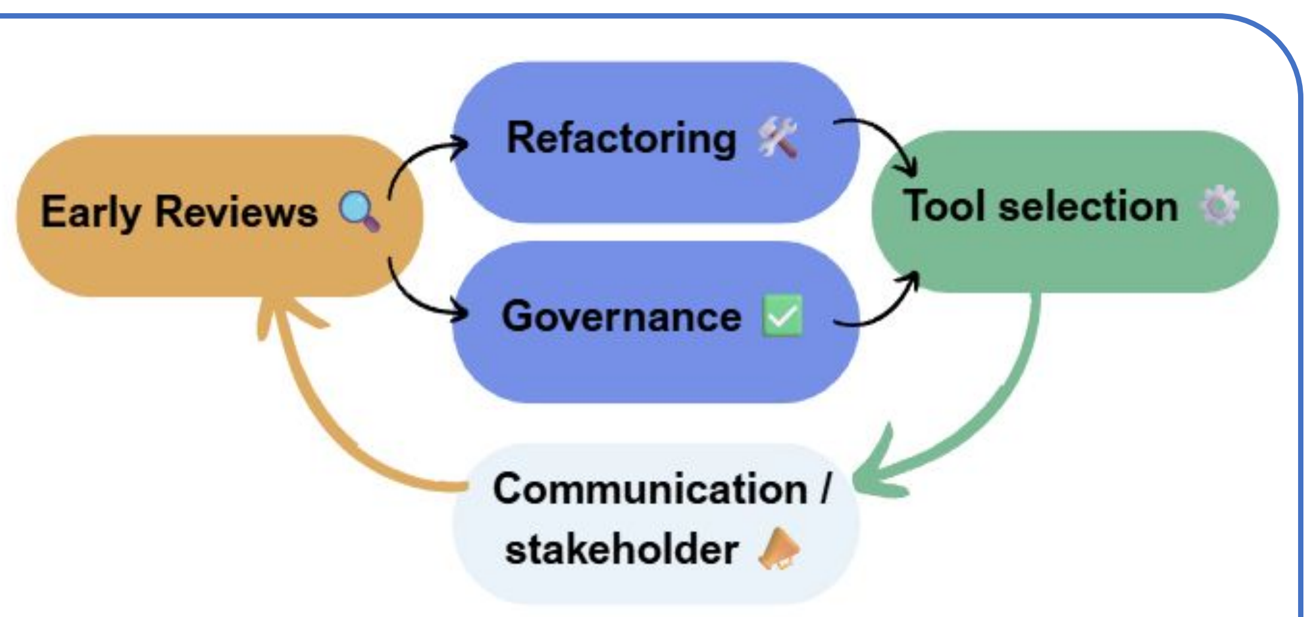
Global Singleton WebDriver

ARCHITECTURAL PAIN
Hidden shared state; parallel runs collide

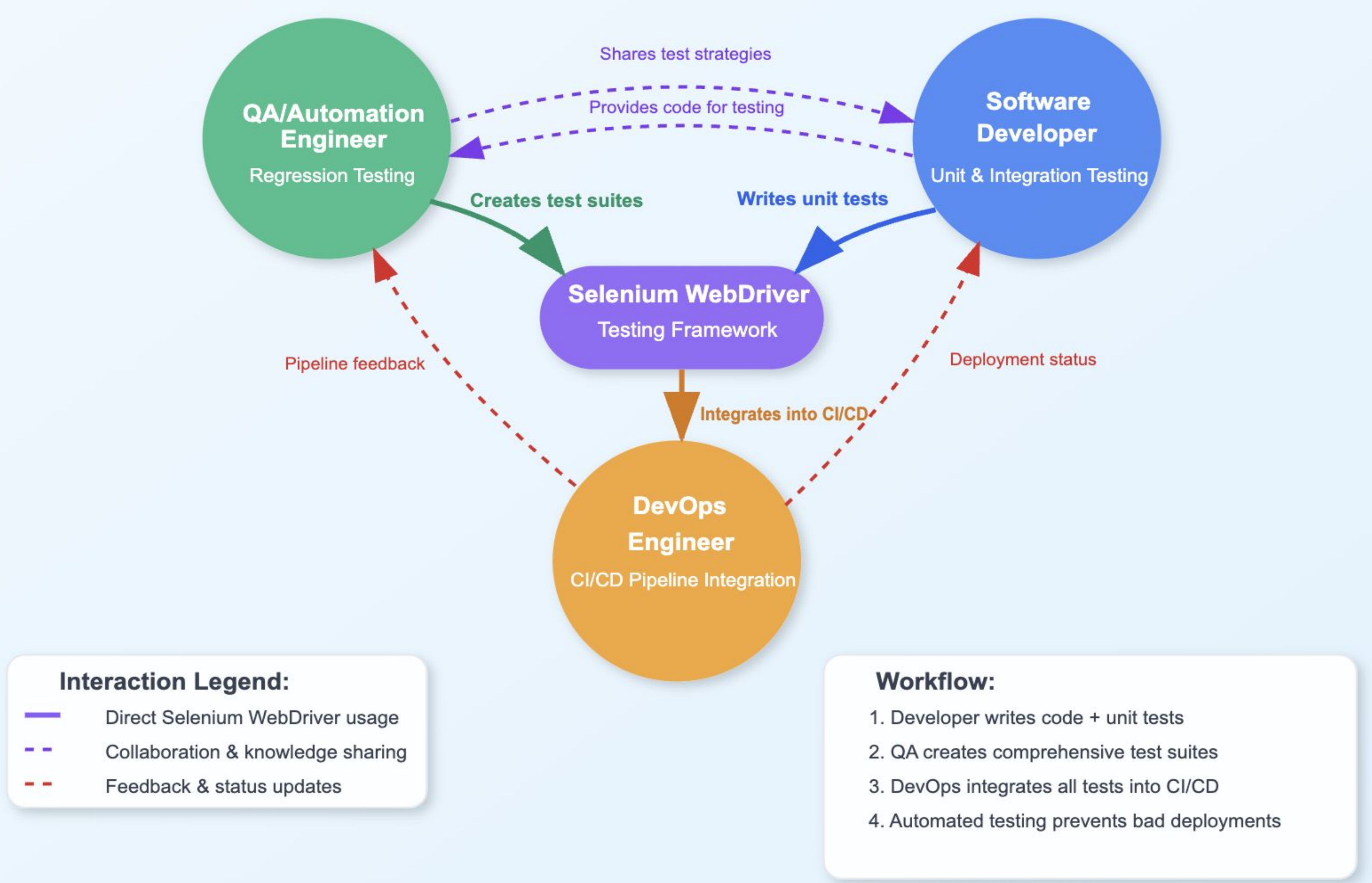
FIX
Provide per-test driver factory; inject & dispose per case

Selenium WebDriver Architecture Improvement Flow

The closed cycle emphasises that architectural-smell insights are applied iteratively to drive modularity, reliability, and rapid deployability.



Actor-Use Case Analysis: Selenium WebDriver Implementation

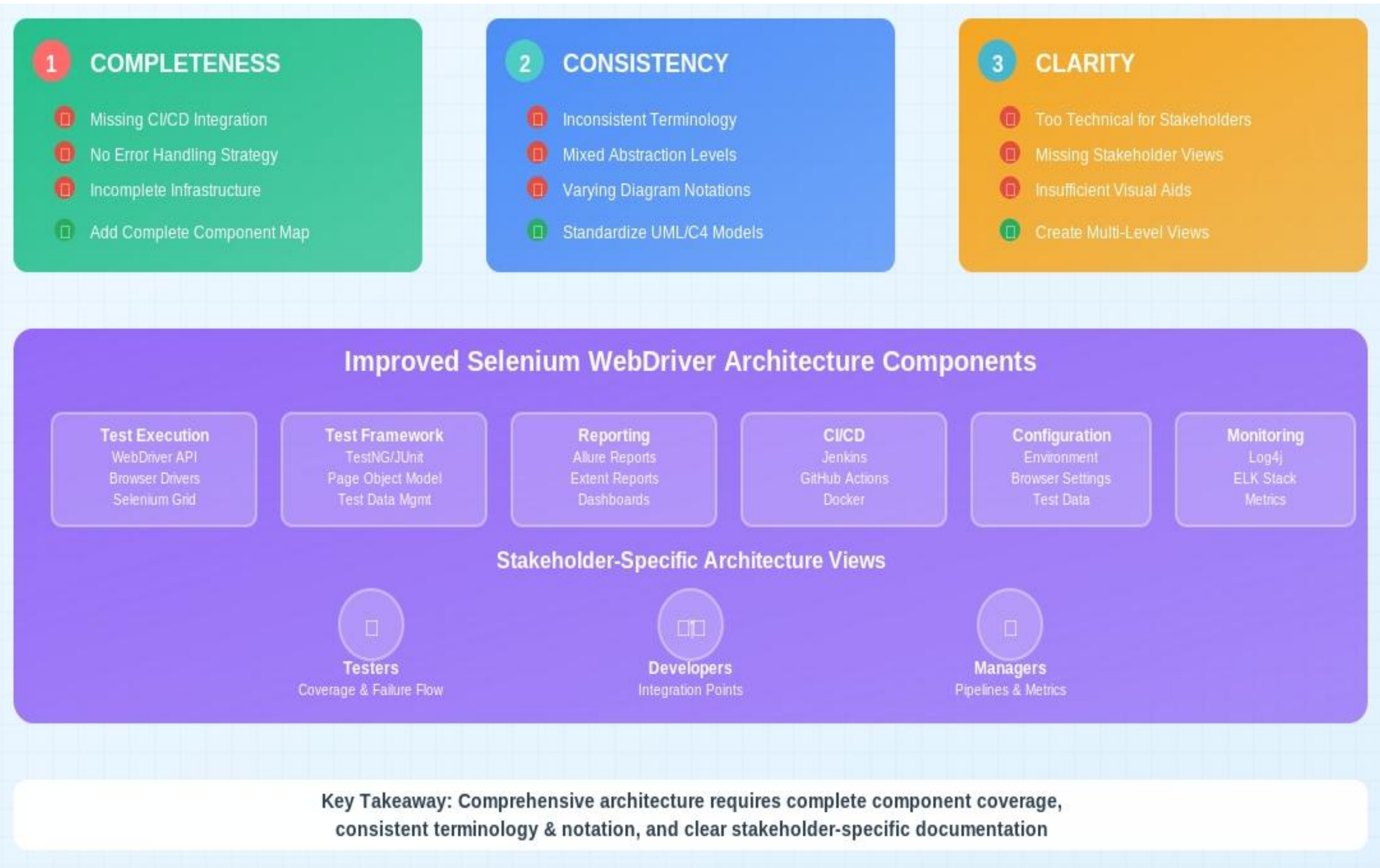


Interaction Legend:
- ━ Direct Selenium WebDriver usage
- - - Collaboration & knowledge sharing
- - - Feedback & status updates

Workflow:
1. Developer writes code + unit tests
2. QA creates comprehensive test suites
3. DevOps integrates all tests into CI/CD
4. Automated testing prevents bad deployments

Selenium WebDriver Architecture Quality Evaluation

Question Framework for Architectural Description Quality



1 COMPLETENESS
- Missing CI/CD Integration
- No Error Handling Strategy
- Incomplete Infrastructure
- Add Complete Component Map

2 CONSISTENCY
- Inconsistent Terminology
- Mixed Abstraction Levels
- Varying Diagram Notations
- Standardize UML/C4 Models

3 CLARITY
- Too Technical for Stakeholders
- Missing Stakeholder Views
- Insufficient Visual Aids
- Create Multi-Level Views

Improved Selenium WebDriver Architecture Components

Test Execution	Test Framework	Reporting	CI/CD	Configuration	Monitoring
WebDriver API	TestNG/JUnit	Allure Reports	Jenkins	Environment	Log4j
Browser Drivers	Page Object Model	Extent Reports	GitHub Actions	Browser Settings	ELK Stack
Selenium Grid	Test Data Mgmt	Dashboards	Docker	Test Data	Metrics

Stakeholder-Specific Architecture Views

Testers	Developers	Managers
Coverage & Failure Flow	Integration Points	Pipelines & Metrics

Key Takeaway: Comprehensive architecture requires complete component coverage, consistent terminology & notation, and clear stakeholder-specific documentation

To improve **Selenium WebDriver** architecture, focus on clarity, completeness, and consistency. Clarity is achieved by using clean, role-based diagrams with clearly labeled interactions between QA, developers, and DevOps. Completeness ensures all essential components—test suites, unit tests, CI/CD integration, parallel execution, and reporting—are represented in the architecture. Consistency involves using standardized terminology, uniform diagram styles, and aligned abstraction levels across documentation. These improvements make the architecture more understandable, comprehensive, and reliable, enabling better collaboration among stakeholders and ensuring alignment between the described architecture and the actual Selenium-based testing framework.

Key Takeaways

- **Applied ISO/IEC/IEEE 42010 with clear patterns, principles, styles, and tactics.**
- **Demonstrated Client-Server & Layered patterns for modular, testable design.**
- **Followed Separation of Concerns & Open/Closed Principle for flexibility.**
- **Used RESTful and Component-Based styles for scalable communication.**