



QA infrastructure in KDE

2026/09/19, Akademy

Bhushan Shah & Nicolas Fella





KDE's biggest problem?

?



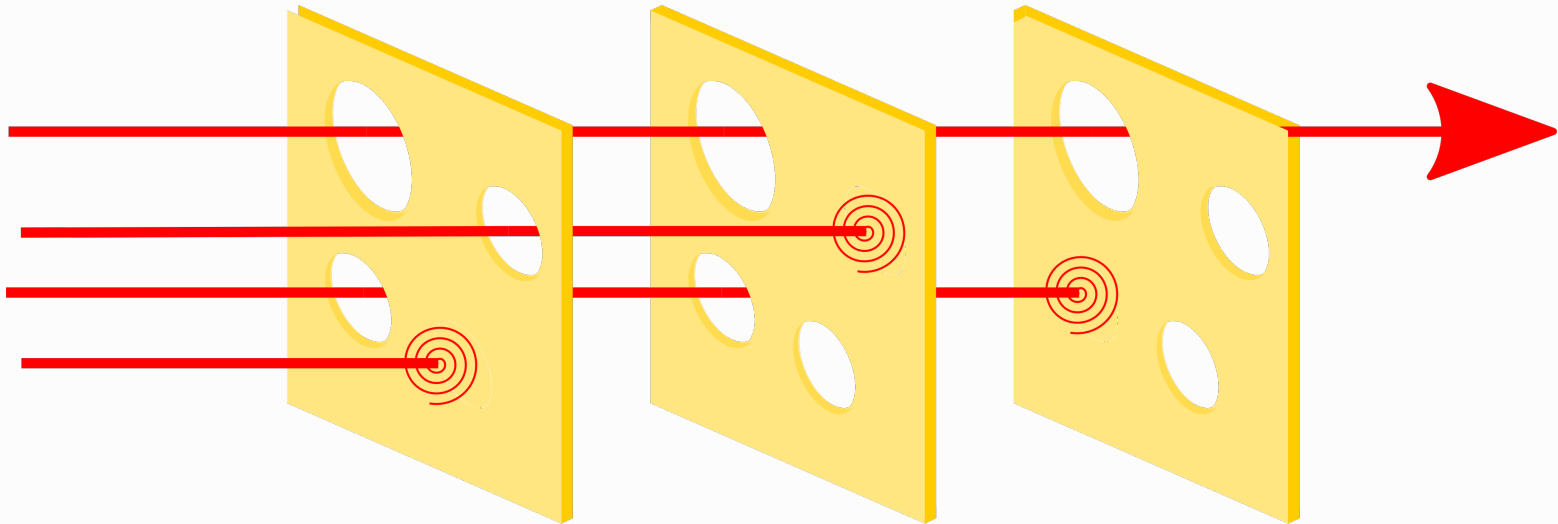
What makes software complicated?

- Lots of functionality
- Options
- Customization (themes, fonts, colors, ...)
- Localization (translation, formatting, RTL, ...)
- Accessibility
- External components
- Hardware



Cheese & Onion

- QA is wider than you think
- Like an onion, it has layers





Scope Management is QA

- Code you didn't write can't have bugs
- Managing scope is managing QA effort



Code Quality is QA

- Well-designed code makes bugs obvious
- Illegal states should be unrepresentable
- Use suitable high-level abstractions/toolkits
- Write expressive, self-documenting code
 - Smart pointers
 - Optional
 - Algorithms
 - Futures



Tools are QA

- Compiler warnings
- Static analysis (clazy, clang-tidy, cppcheck, ...)
- Asserts
- Sanitizers (address, undefined, leak, ...)



Unit tests are QA

- Unit tests test units
- Test logic, parsing, computation,...
- Run during development and on CI
- Test-driven-development
- Not always feasible
- Accounting for tests affects architecture

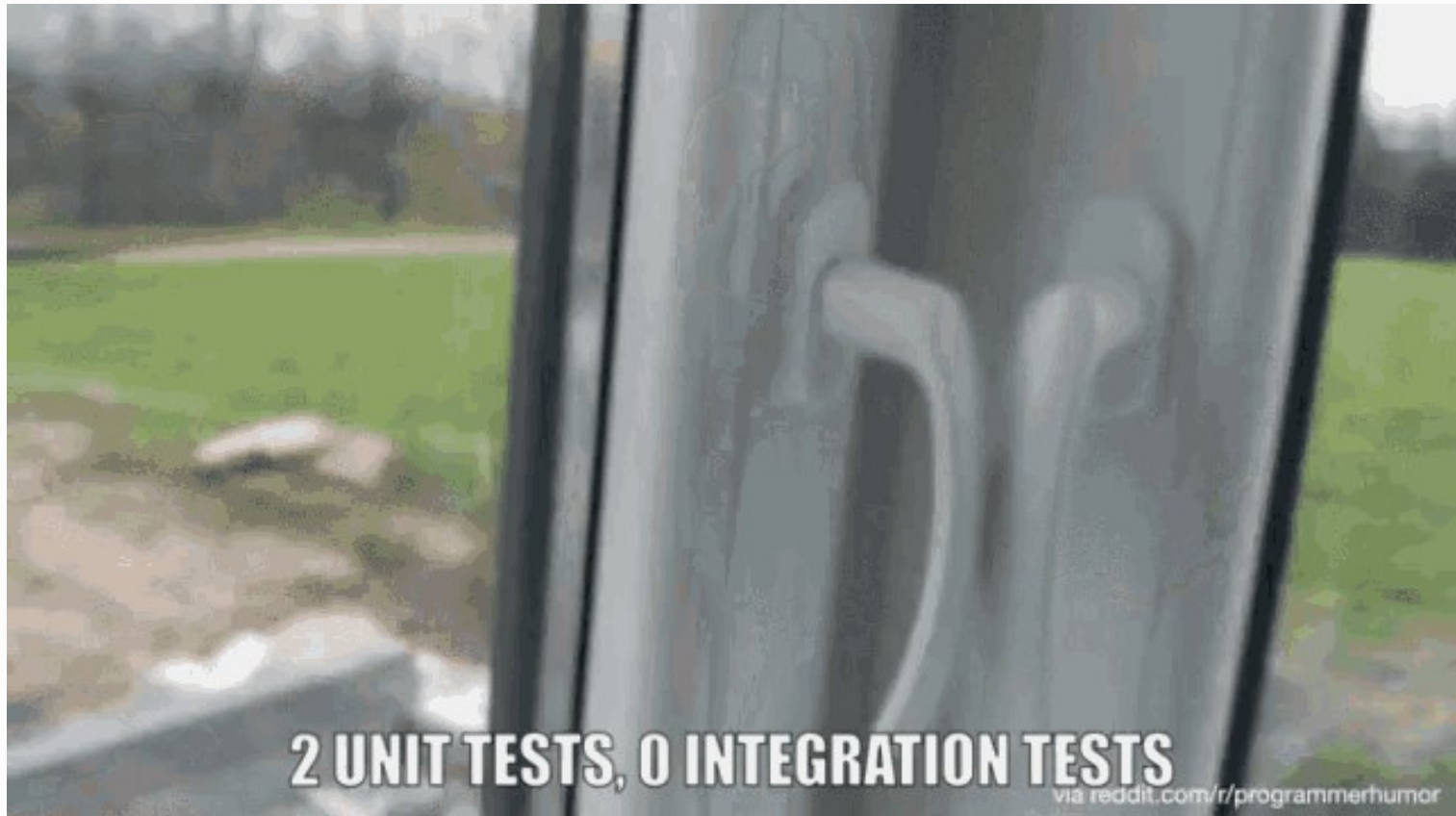


Code Coverage

- Measures how much code is covered by tests
- Different types: function, branch, line coverage
- KDE CI generates coverage report
- Needs to be interpreted carefully



The Problem with Unit Tests





Integration Tests

- Unit tests don't test the full picture
- End-to-end testing is needed



Selenium Webdriver based test cases

- Unit tests are not enough
- Need for testing user interaction
- selenium-webdriver-at-spi is used to translate at-spi API to the to be used by selenium python API
- Selenium can be used to then create the user interaction flows
- Test cases run in nested kwin_wayland session on separate dbus-session



- Selenium based tests are not enough
- Not same environment as how users would use system
- openQA based testing
- Test complete user interaction end-to-end, including installation, upgrade, downgrade etc.
- Prevent visual regressions by needle based testing
- Happens at distribution level



OpenQA – Future plans

- We want to make it possible for components such as frameworks to make use of openQA infrastructure
- Test cases are part of the repository itself, pipeline will start pristine VM with pre-installed KDE Linux and binaries / flatpak installed on top
- Run test cases using pre-defined test workflows



Smoke Tests

- Unit tests/integration tests take effort to develop
- A simple test is better than no test
- Verifies that application/components starts without obvious defects
- Can be used for: Applications, Plasmoids, KCMs, Components



Manual QA

- Automatic testing doesn't catch everything
- Manual testing pre/post merge
- Try to break system
- Testing protocols
- Eat your own dog food
- Should be easy to do



Distribution Tests

- Distributions can/do run integration tests
- Existing OpenQA tests
- Verify the specific system integration



Nightly/Beta channels

- Testing released versions is too late
- Beta releases
- Nightly distributions (KDE Neon, KDE Linux etc)
- Flatpak, F-Droid, ...
- See Volker's talk



Institutional users

- Hardware partners can provide hardware integration QA
- Administrators manage fleet deployments



Shifting Left

- QA spans from development towards end users
- Each step adds delays
- Communication breaks down
- Catching issues earlier is cheaper



Questions?