1. Introduction

Software quality consists of two distinct notions.

Quality is a capability of a software product to conform to requirements.

Functional quality reflects how well it complies with or conforms to a given functional requirements.

Structural quality how well it meets non-functional requirements, such as maintainability and other requirements that support the delivery of functional requirements.

1.1. Bugs

Bug is then software doesn't work as specified.

Types:

Regression type of software functional or performance bug, when previously developed feature stops working. This may happen when changing same code base or changing api(either for new feature or for bug fixes), updating the system or dependencies.

Full acceptance tests can prevent regression.

2. Testing

What to test?

Functional testing answers the question "does this **particular** feature works" or "can the user do this"

End to end answers the question "does this production scenario works" or "can user do this from start to finish". Uses real data and test environment to imitate workload.

Testing parameters:

- Should all corner cases be coverd, few or just happy paths?

How to make test cases:

2.1. Acceptance testing

Communicating requirements may be ambiguous, final acceptance test cases - is the single source of truth, when requirements are done and the work is **done**.

Acceptance testing have the purpose of communication, clarity and precision. By agreeing to them, the developers, stakeholders, and testers all understand what the plan for the system behavior is. Achieving this kind of clarity is the responsibility of all parties. Professional developers make it their responsibility to work with stakeholders and testers to ensure that all parties know what is about to be built.

In general business describes "happy paths", the task of dev is to specify the "corner cases" in test cases.

Every acceptance test should be automated. It's too cost to manually test it each time

Testing via GUI - the layout may be changed for aesthetic reasons, but the underlying capability of the GUI will remain the same. It's much better to write a test that selects the button whose ID is "ok_button" than to select the button in column 3 of row 4 of the control grid.

2.1.1. Who

Developer or QA should write test cases.

If it turns out that developers must write these tests, then take care that the developer who writes the test is not the same as the developer who implements the tested feature. See own mistakes is **harder** than see other's mistakes. Even if developer doesn't write test cases, he can implement them along the main feature.

Unit test describes the low level structure and behaviour of the code. Audience is the programmers. Also unit tests prevents a plethora of bugs, because they are fast ⟶ have fast feedback loop.

Component tests same as acceptance tests(may be them), limited by one component. The components encapsulate business rulles, so the component tests are the acceptance tests for those business rules. It tests output after provided input. Any other components are decoupled from the test using appropriate mocking and test-doubling techniques.

Component tests cover roughly half of the system. They directed towards happy-path and very obvious corner, boundary and alternate-path cases

Example: when side api is not working or when user does not exists, or product does not exists, or user payment is not processed.

2.1.2. Strategy and test automation pyramide

QA and users should find nothing If something found - dev team must prevent it in future.