Quality assurance

Introduction

What

★★☆☆ Can explain software quality assurance

Software *Quality Assurance* (QA) is the process of ensuring that the software being built has the required levels of quality.

While testing is the most common activity used in QA, there are other complementary techniques such as *static analysis, code reviews,* and *formal verification*.

Validation vs Verification

★★★☆ Can explain validation and verification

Quality Assurance = Validation + Verification

QA involves checking two aspects:

1. Validation: are you *building the right system* i.e., are the requirements correct?
2. Verification: are you *building the system right* i.e., are the requirements implemented correctly?

Whether something belongs under validation or verification is not that important. What is more important is that both are done, instead of limiting to only verification (i.e., remember that the requirements can be wrong too).

Exercises

Code reviews

What

★★★☆ Can explain code reviews

Code review is the systematic examination of code with the intention of finding where the code can be improved.

Reviews can be done in various forms. Some examples below:

- **Pull Request reviews**
    - Project Management Platforms such as GitHub and BitBucket allow the new code to be proposed as *Pull Requests* and provide the ability for others to review the code in the PR.
- **In pair programming**

- As pair programming involves two programmers working on the same code at the same time, there is an implicit review of the code by the other member of the pair.

- **Formal inspections**

  - Inspections involve a group of people systematically examining project artifacts to discover defects. Members of the inspection team play various roles during the process, such as:

    - the author - the creator of the artifact
    - the moderator - the planner and executor of the inspection meeting
    - the secretary - the recorder of the findings of the inspection
    - the inspector/reviewer - the one who inspects/reviews the artifact

Advantages of code review over testing:

- It can detect functionality defects as well as other problems such as coding standard violations.
- It can verify non-code artifacts and incomplete code.
- It does not require test drivers or stubs.

Disadvantages:

- It is a manual process and therefore, error prone.

Resources

- **10 tips for reviewing code you don't like** - a blog post by David Lloyd (a Red Hat developer).

Static analysis

What

★★★☆ Can explain static analysis

> **Static analysis**: Static analysis is the analysis of code without actually executing the code.

Static analysis of code can find useful information such as unused variables, unhandled exceptions, style errors, and statistics. Most modern IDEs come with some inbuilt static analysis capabilities. For example, an IDE can highlight unused variables as you type the code into the editor.

The term *static* in static analysis refers to the fact that the code is analyzed without executing the code. **In contrast, *dynamic analysis* requires the code to be executed to gather additional information about the code** e.g., performance characteristics.

Higher-end static analysis tools (static analyzers) can perform more complex analysis such as locating potential bugs, memory leaks, inefficient code structures, etc.

Some example static analyzers for Java: CheckStyle, PMD, FindBugs

***Linters* are a subset of static analyzers** that specifically aim to locate areas where the code can be made 'cleaner'.

Formal verification

What

★★★☆ Can explain formal verification

Formal verification uses mathematical techniques to prove the correctness of a program.

An introduction to Formal Methods

Advantages:

- **Formal verification can be used to prove the absence of errors**. In contrast, testing can only prove the presence of errors, not their absence.

Disadvantages:

- It only proves the compliance with the specification, but not the actual utility of the software.
- It requires highly specialized notations and knowledge which makes it an expensive technique to administer. Therefore, **formal verifications are more commonly used in safety-critical software such as flight control systems**.

Exercises