Software Quality, Verification, and Validation

Oscar Palm

Feb 2023

Contents

Chapter no. 1

Software Quality, Verification, and Validation

1.1 Introduction

1.1.1 Problems in our society

Our society depends on software, cars, water, energy, computers, everything is controlled by software.
Flawed software will hurt profits, fixing a bug after release and delivery is much more expensive than fixing it before its release. Even if bugs aren't noticeable by the end users or the company hosting it, bugs can lead to security exploits, and later breaches.
Flawed software can even hurt users directly, e.g. pacemaker crashing or av making a wrong turn.

1.1.2 Why this course

- What is "good" software?

  we determine this through quality dependencies
- What is the key to good software?s

  Verification and Validation.
- Exploration of testing and analysis activities of the V and V process

1.2 When is software ready for reease?

1.2.1 The short answers

- We can't find any bugs
- When we have finished testing
- When quality is high

1.2.2 The long answer

We all want high-quality software, but be don't all agree what that means.

- We can't find any bugs

  but we can't find all bugs.
- When we have finished testing

  but we can't test everything.

- When quality is high

  but we don't know what that means.

We need to define what we mean by quality, and how we can measure it.

Quality attributes

- Performance

  Ability to meet timing requirements

- Security

  Ability to protect information

- Scalability

  Ability to grow the system to process more concurrent requests

- Availability

  Ability to carry out the task whenever needed

- Modifiability

  Ability to enhance the software to meet new requirements or fix bugs

- Testability

  Ability to easily find faults in the software

- Interoperability

  Exchange information with other systems

- Usablility

  Ability to be used by the intended audience and perform required tasks

  How easy it is to learn to use the software

These can easily conflict with each other, its important to decide what to prioritize and set a threshold of what's good enough.

1.2.3 When is software ready for release?

It's ready for release when it's **dependable**. This means it needs to be correct, reliable.

1.3 Verification and Validation

1.3.1 Verification

The process of checking that a system meets its requirements.

Are we building the product right?

Verification is an experiment. We perform trials, evaluate the results, and gather information about what and why it happened.

Testing

An investigation into system quality, it's based on sequences of stimulations and observations. The software version of a lab rat we later dissect to analyze what failed.

1.3.2 Validation

The process of proving that it meets the specifications set by the customer.

Are we building the right product?

Does the product work in the real world? Even if the software does exactly what we set out to do, it might not be what the customer wants.

1.3.3 Conclusion

Verification checks if the software works as intended.
Validation checks that the software is useful. (This is much harder)

Both are important and complete each other. This class however, focuses largely on verification.

- Testing is the primary activity of verification.

1.4 Required level of V and V

Depends on:

- Software Purpose

  The more critical, the more important that it works

- User Expectations

  Some users are more forgiving than others

- Marketing environment

  With competing products in a market, it might be more important to release a product quickly than to make it perfect.

1.4.1 Basic questions

1. When do verification start and end?
   - It should start as soon as the project starts

     we need to know what we're building and how design/technical choices affect our product's quality

     A great starting point is static verification
   - It ends when the product is released

     A great way of verifying during the development process is through dynamic verification
2. How do we obtain an acceptable level of quality at an acceptable cost?
3. How do we decide when it's ready to release?
4. How can we control quality during the development process?

1.5 Trade offs

There's always a trade-off when designing software, "Better, faster, or cheaper - pick any two".

1.5.1 Verification Trade-offs

We are interested in proving that a program demonstrates property X

- Pessimism inaccuracy

  Not guaranteed to program even if X is true

- Optimism inaccuracy

  May be true, even if X is false

- Property Complexity

  if X is too difficult to check, substitute with simpler property Y

Finding all faults is nearly impossible, instead we need to decide ourselves when we are ready for release, how good is good enough?

We need to establish criteria for what is good enough, and what is not. One way of doing this is through **Alpha/Beta testing** where a small group of users gets the chance to use the product in a somewhat controlled environment and reports feedback and failures.