EECS4313 week 1

Jerry Wu

2024-01-08

Contents

Lecture 1

Introduction

Testing knowing and unknowing

For knowing, we already have an idea of the features we need to implement.

1. Requirements
2. Deliverable

The opposite being testing stuff that we don't know we want to implement or the requirements don't cover(extrapolating new features). Oftentimes, the base requirements aren't enough to solve the problem that the software wants to solve.

Our goal is to find unknowing functional/unfunctional bugs and problems within code effectively.

Difference between testing goals

- An oracle is used for known testing, because we can get a definitive and objective answer as to whether something fulfills the requirements.
- Testing the **unknown** requires subjectivity and whether something is good or bad, which we will focus on in this course

Common causes of software failure

To put simply, a software failure is any kind of unpredictable/undesirable behavior. Some examples include:

- Crashing (the worst case scenario)
- Hanging
- Data corruption
- Incorrect functionality
- Performance degradation

Our goal is to completely prevent these issues and in the worst case scenario where they can't be prevented to recover or arrive at a termination point so as to create a functional system.

A few more possible causes

- Segfault (index out of bounds)
- Deadlock (in multithreaded applications with thread interaction over shared memory)
- Memory leaks
- Regression bugs (fixing one bug creating multiple new bugs)
- Incorrect implementation
  for example:

```
1 public int add(int x, int y)
2 {
3     return x-y
4 }
```

Historical examples of bugs

- Ariane 5 rocket crash (64 bit float for horizontal velocity converted to 16 bit signed int. Number was $>$ 32767, which is the largest 16 bit int, conversion failed)
- This ended up costing $8B