Program Inspections and Walkthroughs

Introduction

- Software testing SHOULD NOT be confined to testing the **running** software.

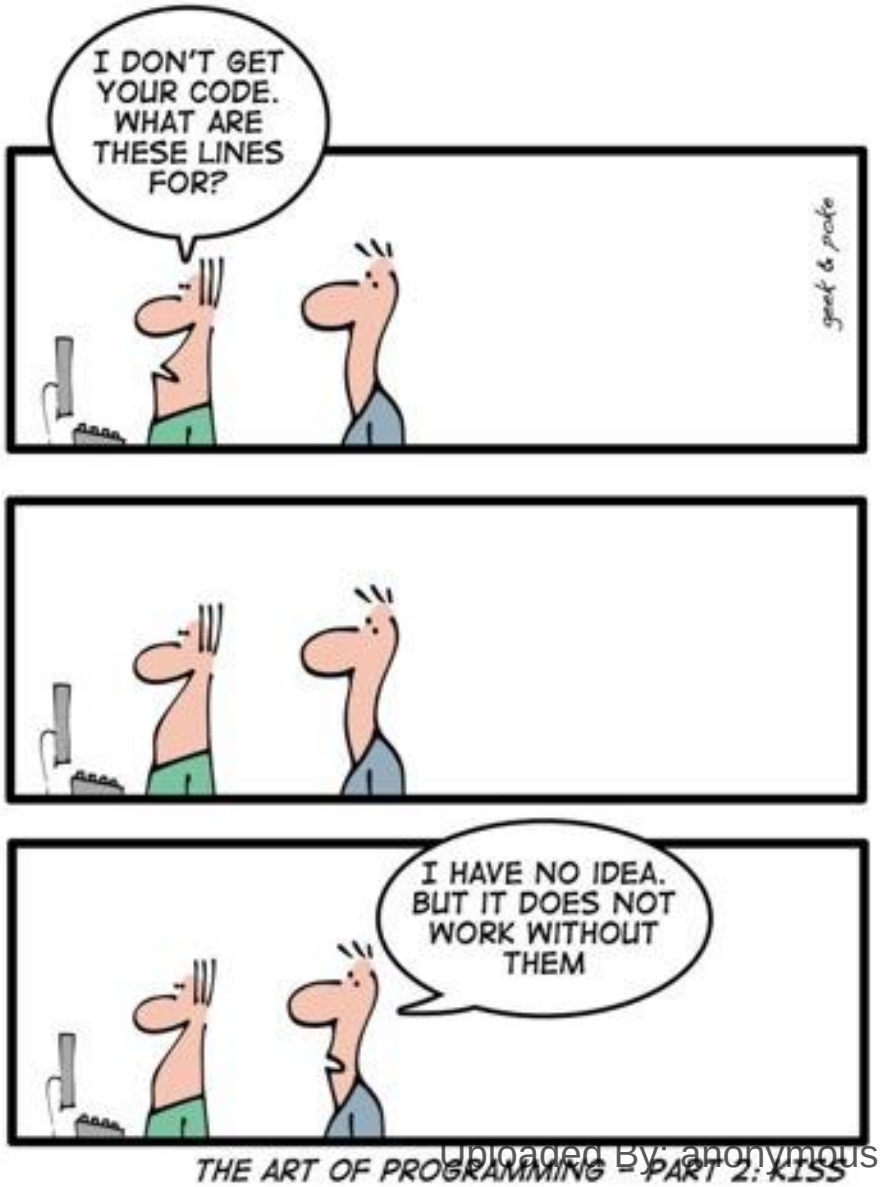
- Scientific research has proven the great value of **reading** code as part of a comprehensive testing and debugging regimen.

Introduction..2

- Human testing techniques are quite effective in finding errors—so much so that every programming project should use one or more of these techniques.
- Research has proven that
  - The earlier the errors are found
    - The lower costs of correcting them;
    - The higher probability of correcting them correctly

Factors that Influence the use of Code Inspections

- **Size** and **complexity** of application
- Size of **development team**
- **Experience** of development team
- **Timeline** of project
- The **culture** and background of team (are they professional, or will they take it personally?)



Inspections and Walkthroughs

- Inspections and walkthroughs involve a team of people
  - **reading**
  - **or visually inspecting a program.**
- The objective here is find **errors**.
- In a walkthrough, a group of developers—with three or four being an optimal number—performs the review.
- Only **one** of the participants is the author of the program.
- Therefore, the majority of program testing is conducted by people other than the author

Inspections and Walkthroughs

- Another advantage of walkthroughs, resulting in lower debugging (error-correction) costs, is the fact that when an error is found, it usually **located precisely** in the code, as opposed to black box testing, where you only receive an unexpected result.

- These human testing methods generally are effective in finding from **30 to 70** percent of the logic-design and coding errors in typical programs

- Inspections & Walkthroughs + Computer-based testing **Complementary**
- Highly applied at **Google and Mozilla**!

Inspection Team: The Moderator

- He is a skilled programmer
- responsibilities are to:

- Distributing materials for, and scheduling, the inspection session.
- Leading the session.
- Recording all errors found.
- Ensuring that the errors are subsequently corrected.

Inspection Team

- **Programmer**: the one who wrote the code
- **Team leader**: senior programmer/team lead
- **Test Specialist**:
  - The specialist should be well versed in software testing
  - and familiar with the most common programming errors

Inspection Process

1. Distribute code listing several days before session
2. During session, the programmer narrates code statement by statement
3. The program is analyzed against a pre-defined set of common programming errors
4. At the end, the programmer is given a set of errors
5. If errors require substantial corrections, plan for another session after correction
6. The list of errors are analyzed, categorized and saved for future inspections

Common Programming Errors

- Does a referenced variable have a value that is unset or uninitialized?
- For all array references, is each subscript value within the defined bounds of the corresponding dimension?
- When indexing into a string, are the limits of the string off by one in indexing operations or in subscript references to arrays?
- Each variable is correctly initialized/proper value?
- Are there any mixed-mode operations?
- Is it possible for the divisor to be ZERO?

Common Programming errors..2

Are the operands of a Boolean operator Boolean? Have comparison and Boolean operators been erroneously mixed together? This represents another frequent class of mistakes. Examples of a few typical mistakes are illustrated here:

- If you want to determine whether *i* is between 2 and 10, the expression `2<i<10` is incorrect. Instead, it should be `(2<i)&&(i<10)`.
- If you want to determine whether *i* is greater than *x* or *y*, `i>x||y` is incorrect. Instead, it should be `(i>x)||(i>y)`.
- If you want to compare three numbers for equality, `if(a==b==c)` does something quite different.
- If you want to test the mathematical relation `x>y>z`, the correct expression is `(x>y)&&(y>z)`.

Common Programming Errors..3

- Will every loop eventually terminate?
- Will every loop execute in the first place?
- Loop boundaries:

For example, if you want to create Java code for a loop that iterates 10 times, the following would be wrong, as it performs 11 iterations:

```
for (int i=0;i<=10;i++){
    System.out.println(i);
}
```

Common Programming Errors..4

- Writing too much code inside methods.
- Incorrect use of inheritance in Object-Oriented programming.
- Not checking methods parameters' data before processing them
- Incorrect exception handling:
  - Catching for no reason
  - Hiding/ burying exceptions.
  - Propagating exceptions
- Forgetting to close/release system resources such as DB connections and files

Walkthroughs

- The initial procedure is identical to that of the inspection process:
  - The participants are given the materials several days in advance to allow them time to bone up on the program
- However, the difference is that rather than reading the code line by line,
-
- ***Participants play computer and execute special test cases***
- ***The state of program is monitored on whiteboard***
- ***Test cases serve as vehicle to ask the programmer questioning***
- ***More errors can be revealed by asking***

Desk Checking

- A desk check can be viewed as a one-person inspection or walkthrough.
  - A person reads a program, checks it with respect to an error list,
  - and/or walks test data through it.
- Better performed by a person other than a programmer.
- Not very effective, **WHY?**

Peer Rating

- Peer rating is a technique of evaluating anonymous programs in terms of their overall
  - quality,
  - maintainability,
  - extensibility,
  - Usability,
  - and clarity.
- The purpose of the technique is to provide programmer self evaluation.

Peer Rating Process

1
- The admin selects 6-20 participants of similar backgrounds
- Each participant is asked to provide two programs: one high quality and another with lower quality
- Session = 30 minutes

2
- Each participant is randomly assigned 4 programs: 2 high and 2 low quality
- Note that the process is totally anonymous

3
- Each participant is asked to give feedback (scale 1-10), about:
- Clarity, readability, design, maintainability, any comments?

4
- Feedback is given to programmer along with their resulting ranking

OBJECTIVES SUMMARY

The direct objectives are:

- To detect analysis and design errors.
- To identify new risks expected to affect the completion of the project.
- To identify deviations from templates and style procedures.
- To approve the analysis or design product, allowing the team to continue to the next development phase.

The indirect objectives are:

- To serve as an informal meeting place for the exchange of knowledge about development tools, techniques, experience with new tools, methods and related items.
- To promote and support the improvement of development methods by supplying new data for analysis of design errors.

Error Handling Techniques

- Research has proven that a large percentage of software errors are caused by **bad error handling**
- This is critical when there are **novice** programmers in your team.
- Usually, error handling is considered to be an **advanced** task to program.
- Keep **a good eye** on error handling techniques applied, and you will be surprised at what you can find there!

Error Handling Techniques..2

1. Return a neutral value:
   - Sometimes the best response to bad data is to continue operating and simply return a value that's known to be harmless

- 2. Substitute the next piece of valid data
  - Example reading from a database or reading from a thermometer sensor in a mobile app.

Error Handling techniques..3

- 3. Return the same answer as the previous time:
  - Example: calculating velocity and reading from sensors.
- 4. Substitute the closest legal value.
  - Example: temperature should only be between 0-100 C.
- 5. Log an error message.
- 6. Return an error code:
  - Example: reading a record that does not exist from a database.
  - You can throw a specific exception if you want.
- Shut down:
  - Applied in critical systems.

Exceptions Best Practices

- If a method encounters an error that does not know how to handle it, it **can throw an exception**
- It is like saying, "*I don't know what to do about this—I sure hope somebody else knows how to handle it!*"
- **Use exceptions to notify other parts of the program about errors that should not be ignored**
- **Throw an exception only for conditions that are truly exceptional.**

Exceptions Best Practices..2

- **Throw exceptions at the right level of abstraction**
- **Include in the exception message all information that led to the exception.**
- **Avoid empty catch blocks**
- **Should have centralized exception handling techniques.**

```
class Employee {
   ...
   public TaxId GetTaxId() throws EOFException {
      ...
   }
   ...
}
```

Bad Java Example of Ignoring an Exception

```
try {
   ...
   // lots of code
   ...
} catch ( AnException exception ) {
}
```

```
try{
    .....
    .....
catch(SomeException ex){
    throw ex;
}
```

Tutorial 6: General Code Inspection Guidance

- **Safe from bugs:**
  - Correct today and correct in the unknown future.
- **Easy to understand:**
  - Communicating clearly with **future** programmers, including **you**
- **Ready for change:**
  - Designed to accommodate **change** without rewriting.

Tutorial 6 Solution

- **Do Not Repeat Yourself (DRY)**
  - Duplicating bugs.
  - Nightmare for maintenance team
  - How can you DRY it out?
- **Comments**.
  - How good or bad is it commented?
- **Different ways of calling the method problem.**
  - There are other programmers than you!
- **The problem of Magic numbers**
  - Never **hard-code constants** that you calculated by **hand**
- **Do not reuse parameters!!**