CMPE 131
Software Engineering

October 12, 2017

GitHub and Software Testing

Presented By

Melvin Ch’ng

San José State
UNIVERSITY

Agenda

- GitHub and Demo
- Unit Testing
- Component Testing
- Testing Demo
- Q&A
- Interesting Video

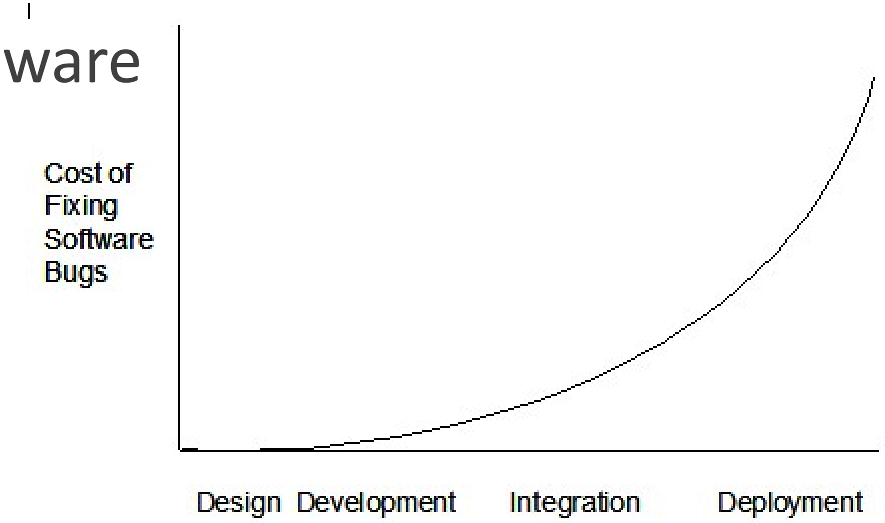
GitHub

- GitHub version control tutorial
  - https://guides.github.com/
    - https://guides.github.com/activities/hello-world/
    - https://guides.github.com/introduction/getting-your-project-on-github/
- Melvin’s version control tutorial
  - http://melvinchng.github.io/rails/VersionControlGithub.html

GitHub Demo

Software Testing

- How do you know if something works?
  - Test it by yourself
  - Write a script to test your program
- Identify bugs (unexpected feature)
- Reduce cost of fixing software
  - Design vs Deployment stage



Unit Testing

- Test individual unit, classes, or methods
- More like a White Box Testing
  - Internal structure, design, or implementation is known to tester
- Contains short code fragments
  - May contain comments
    - People that work with you may not know the language that well

Unit Testing, *cont’d*

- Example C++ functions
  - Add two values
  - Subtract two values

```
// Some unrelated code

float add_two_values(float x, float y){
    return (x+y);
}

float subtract_two_values(float x, float y){
    return (x-y);
}

// Some unrelated code
```

Unit Testing, *cont’d*

- Check if the values return from the function is correct

```
// Some unrelated code

float a[6] = {3.5, 4.2, 3.4, 5.2, 2.1, 1.1};
float b[6] = {1.2, 9.1, 2.4, 0.9, 1.8, 1.7};

for(int i = 0; i < 5; i++){
    if (add_two_values(a[i], b[i]) != (a[i] + b[i])){
        cout << "Error adding " << a[i] << “&” << b[i];
    }
}

cout << "Completed Add Test" << endl;

// Some unrelated code
```

Component Testing

- Test a specific module in a program
- A component may
    - contains a few individual units, classes, or method
    - contains a few different components
    - Depend on other modules
    - May contain comments
        - People that work with you may not know the language that well
- Usually perform after a successful unit testing

Component Testing, *cont'd*

- Example C++ functions to add and subtract two values

```
// Some functions

float calculations(float a, float b){
  return add_two_values(a, b)+subtract_two_values(a, b);
}

// Some functions
```

Component Testing, *cont'd*

- Check if the values return from the function is correct

```
// Some unrelated code

float a[6] = {3.5, 4.2, 3.4, 5.2, 2.1, 1.1};
float b[6] = {1.2, 9.1, 2.4, 0.9, 1.8, 1.7};

for(int i = 0; i < 5; i++){
    if (add_two_values(a, b)!= (a[i]+b[i])+(a[i]-b[i])){
        cout << "Error adding " << a[i] << “&” << b[i];
    }
}

cout << "Completed Add Test" << endl;

// Some unrelated code
```

Result of Testing

- Success if input and output are expected
  - Numerical data (integer or float) are used for math operations
  - Identify non numerical data are used for math operations
- Fail if input is expected but output is unexpected
  - Output of adding two values together is not expected
- Brief explanation why a test is successful or unsuccessful

Sample Unit and Component Test code

- C++
  - Source code
    - https://gist.github.com/melvinchng/babdb45fd2a89449a7397dc1f9324486
  - Online C++ compiler
    - http://cpp.sh/
- Ruby
  - Source code
    - https://gist.github.com/melvinchng/3c1df2a0bf52960662e408f9eaa9d25b
  - Online Ruby compiler
    - https://repl.it/languages/ruby

Q&A

Interesting Video