Tools and Process for Quality Code

1

Automate, Automate, Automate

- Unit, Integration, Functional Tests
- Code Coverage reports
- Static Analysis tools
- Code Complexity reports
- Peer Review
- Capture historical metrics

2

Code Coverage

- How thoroughly is your production code exercised by your test code
- Indicator as to the quality of your **tests**, not necessarily prod code

3

Function coverage

- function (method) invoked

```
int foo (int x, int y)
{
    int z = 0;
    if ((x>0) && (y>0)) {
        z = x;
    }
    return z;
}
```

- satisfied by foo(1,1)

4

Line coverage

- line of code executed

```
1  int foo (int x, int y)
2  {
3      int z = 0;
4      if ((x>0) && (y>0)) {
5          z = x;
6      }
7      return z;
8  }
```

- satisfied by foo(1,1) - every line (including z=x) gets hit

5

Decision Coverage

- all code paths executed

```
1  int foo (int x, int y)
2  {
3      int z = 0;
4      if ((x>0) && (y>0)) {
5          z = x;
6      }
7      return z;
8  }
```

- satisfied by foo(1,1) and foo(0,1)
  - foo(1,1) goes into the if statement
  - foo (0,1) does not go into the if statement

6

Condition Coverage

- each boolean sub-expression evaluated

```
1  int foo (int x, int y)
2  {
3      int z = 0;
4      if ((x>0) && (y>0)) {
5          z = x;
6      }
7      return z;
8  }
```

- satisfied by foo(1,1), foo(1,0) and foo(0,0)
  - foo(1,1) satisfies both (x>0) && (y>0)
  - foo(0,1) satisfies (y>0)
  - foo(1,0) satisfies (x>0)

7

Code Coverage caveats

- good code coverage doesn't imply good tests
- most useful for finding gaps in test coverage

8

Java Tools

- Emma
- Cobertura
- Clover (commercial)

9

Grails

- grails install-plugin code-coverage

10

Static code analysis

- Analyze code (in an automated and repeatable fashion) to find
  - defects
  - bad practices
  - inconsistencies
  - style issues

11

What to check

- Largely project / team dependent
- Quick Hit examples:
  - logging: println, system.err, system.out
  - concurrency issues: SimpleDateFormat property should not be static
  - testing: JUnit SetUp method should call super

12

Tools

- Java
  - PMD, Checkstyle, FindBugs
- Groovy
  - CodeNarc, GMetrics
- .NET
  - VS 2010 **Premium**, FxCop, StyleCop

13

UI Static Analysis

- JavaScript is code, too
- As sites become more and more interactive on the front end, lots of functionality goes into the UI
- Can help with cross browser compatibility issues

14

Tools

- JavaScript
  - JSLint (http://jslint.com/)
  - Google Closure Compiler (https://developers.google.com/closure/compiler/)
  - JSHint (http://www.jshint.com/)

15

JSLint and Grails

- grails install-plugin jslint
- grails jslint

- create a config file grails-app/conf/JsLintConfig.groovy to customize settings

16

Continuous Integration

- Frequently integrate code from all developers
- Build, compile and test
- Builds triggered by changes in repository (not on a schedule)
- Notifications

17

Maintain Code Repository

- Use source control
  - Subversion
  - Git
  - Mercurial
  - TFS
  - tons of options - pick one and use it!
- Everything required to build and test the system should reside in source control
- Designate a branch that is the current "main line" (e.g. trunk, master, etc) that should ALWAYS build successfully

18

Automate the Build

- build should be reproducible
- include database schema
- analyze changes and perform appropriate actions based on changeset
  - you don't have to rebuild a component if it hasn't changed
- keep it fast

19

Self Testing

- include automated tests in the build process
  - unit, integration, functional
- static analysis
- if possible, test in a clone of production

20

Public, visible, and easy to use

- make it easy for anyone to get the latest build artifacts
- publish results for all to see
  - build (lava) lamps
  - information radiator (tv screen / old computer monitor)
- send notifications to entire team

21

Build Tool - Jenkins

- Formerly known as Hudson
- Extensible build server
- Web Based

- Master/Slave capabilities for distributed build systems
- Plugin architecture (over 400 plugins)

22

Peer Review

- Another form of (human based!) static analysis
- Knowledge sharing
- Peer pressure to write good code
- Find bugs early

23

Example Peer Review process

- Small team, 5 developers
- no dedicated QA
- "Master" branch pushed to production several times a week

24

Feature Branches

- Start a task, create a new branch
- development occurs on that branch by a single (or multiple) developers
- Jenkins job monitors the repository for new / deleted feature branches, creates a build for each feature branch
- Feature branches can be automatically deployed to a test server

25

Peer Review

- When feature implementation is complete, developer issues a "pull request" to the master branch
- Reference any bug tickets fixed
- Document what the feature is, how it works, and proposed test strategy
- A different developer(s) picks up the task of reviewing the pull request

26

Peer Review

- Double check the build(s) for the branch are all passing
- Deploy the feature branch to a test server and try to break it

- Review the code, offer feedback/comments on the changes made in the branch
- in 2 months of working with this process:
  - 136 pull requests
  - 67% were revised before merge into master based on review/testing

27

Peer Review

- Automated cleanup
- When branch is merged into master
  - builds created for the branch are deleted
  - test server is freed up for a new feature branch
- Master branch is built (and deployed to production)

28

Project health over time

- Statistics can help indicate trouble spots
- Record and report on stats over time
  - number of tests
  - test coverage
  - code complexity
  - static analysis bugs
- Sonar (http://www.sonarsource.org/)

29

References

- http://www.javaranch.com/journal/2004/01/IntroToCodeCoverage.html
- http://en.wikipedia.org/wiki/Static_code_analysis
- http://blog.octo.com/en/analyzing-groovy-grails-code/
- http://stackoverflow.com/questions/534601/are-there-any-javascript-static-analysis-tools
- http://martinfowler.com/articles/continuousIntegration.html