Code Review

Lecture 7
GSL Peru 2014

Why Code Review?

- Find and fix mistakes
- Enforce coding standard
- Improves overall quality
- More eyes, the better!

RANGES OF DEFECT REMOVAL EFFICIENCY

	Lowest	Median	Highest
1 Requirements review (informal)	20%	30%	50%
2 Top-level design reviews (informal)	30%	40%	60%
3 Detailed functional design inspection	30%	65%	85%
4 Detailed logic design inspection	35%	65%	75%
5 Code inspection or static analysis	35%	60%	90%
6 Unit tests	10%	25%	50%
7 New Function tests	20%	35%	65%
8 Integration tests	25%	45%	60%
9 System test	25%	50%	65%
10 External Beta tests	15%	40%	75%
CUMULATIVE EFFICIENCY	75%	98%	99.99%

Types of Code Review

- Formal Inspections (Fagan)
- Over-the-Shoulder
- E-mail Pass Around
- Tool-Assisted
- Pair Programming

Formal Inspection

- 3-6 participants
- Moderator/Controller - organizer
- Prepare material before meeting
- Roles
  - Reviewer - critical analysis
  - Observer - domain expert
  - Reader - reviews for comprehension

Over-the-Shoulder

- Common and Informal
- As name implies, reviewer stand over the shoulder
- Easy to implement
- Roles
  - Author
  - Reviewer

E-mail Pass Around

- Packaged source code for review is emailed
- Second most common - open source
- Easy to implement - cumbersome to incorporate feedback

Tool-Assisted

- Automates process
- Review enforcement
- IDE integration

Examples

CodeCollaborator, github, etc.

Pair Programming

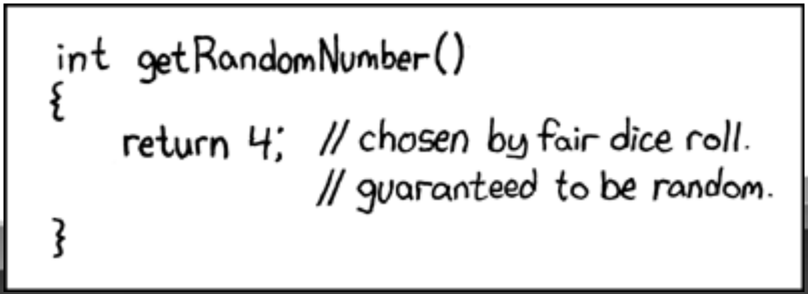
- Part of eXtreme Programming (XP)
- Effective in defect finding and knowledge transfer
- Doubles resources in development

Static Code Analysis

- Detect errors in the code
- Enforce Coding Standard
- Metric Computation

Detect Errors in Code

- Detect Errors
    - array and string handling
    - operation priority
    - memory overruns
    - etc
- Vulnerbility Analysis
- Doesn’t Detect Logic errors



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

From xkcd

Metric Computation

- Lines of Code (LOC)
- Number of empty lines
- Number of comments
- Percent of comments (ratio of the number of lines containing comments to the general number of lines represented in percent)
- The average number of lines for functions (classes, files)
- The average number of lines containing source code for functions (classes, files)
- The average number of lines for modules
- etc

Example

```
bool clearString(char *str)
{
    memset(str, 0, sizeof(str));


    return true;
}
```

Example - check input values

```
void clearString(char *str)
{
    if (NULL != str)
    {
        size_t len = strlen(str);
        memset(str, 0, len);
    }
}
```

Example

```
int findResult(int x)
{
    lock(key);

    if (0 == x)
        return x;

    if (0 == x%2)
        return 2;

    unlock(key);

    return -1;
}
```

Example - single entry, single exit

```
int findResult(int x)
{
        int returnValue = -1;

        lock(key);

        if (0 == x)
                returnValue = 0;
        else if (0 == x%2)
                returnValue = 2;

        unlock(key);

        return returnValue;
}
```

Example

```
<?php

    $out = $a > 9 && $a < 15 ? "option1" : $a < 5
? "option2" : "option3";

?>
```

Example - operator precedence

```
$out = $a > 9 && $a < 15 ? "option1" : $a < 5 ? "option2" : "option3";
```

=> “option1” ? “option2” : “options3”

```
$out = (
    ($a > 9 && $a < 15)
    ? ("option1")
    : (
        ($a < 5)
        ? ("option2")
        : ("option3")
    )
);
```

Example

```
<?php
function foo() {
  $val[1] = 2;
  $val[2] = 3;
  $val[3] = 4;
  $val[4] = 5;
  $val[5] = 6 ;
  $val[5] = max;
  return val;
}
?>
```

Example - double assignment / unassigned value

```
<?php
  function foo() {
    $max = 7;
    return array(2, 3, 4, 5, 6, $max);
  }
?>
```

Example

```
$find = str_replace(",", "", $find);
$find = str_replace(".", "", $find);
$find = str_replace("/", "", $find);
$find = str_replace(" ", "", $find);
$find = str_replace("-", "", $find);
$find = str_replace("+", "", $find);
$find = str_replace("#", "", $find);
```

Example - Poor use of function

```
$ignore = array(",", ".", ".", “ “, “-”, “+”, “#”);
$find = str_replace($ignore, "", $find);
```

Example

```
if (1 || $o->checkForStatus()) {
    return $o->getId() ;
    return $o->getSize() ;
}
```

Example - Linked List

http://www.codediesel.com/php/linked-list-in-php/