

---

# Code Review

CS580 Advanced Software Engineering

<http://cs580.yusun.io>

October 29, 2014

Yu Sun, Ph.D.

<http://yusun.io>

[yusun@csupomona.edu](mailto:yusun@csupomona.edu)



---

CAL POLY POMONA

---

# What is Code Review?

---

## Code Review

is the systematic examination by one's peers of computer source code intended to find and fix mistakes overlooked in the initial development phase.<sup>1</sup>

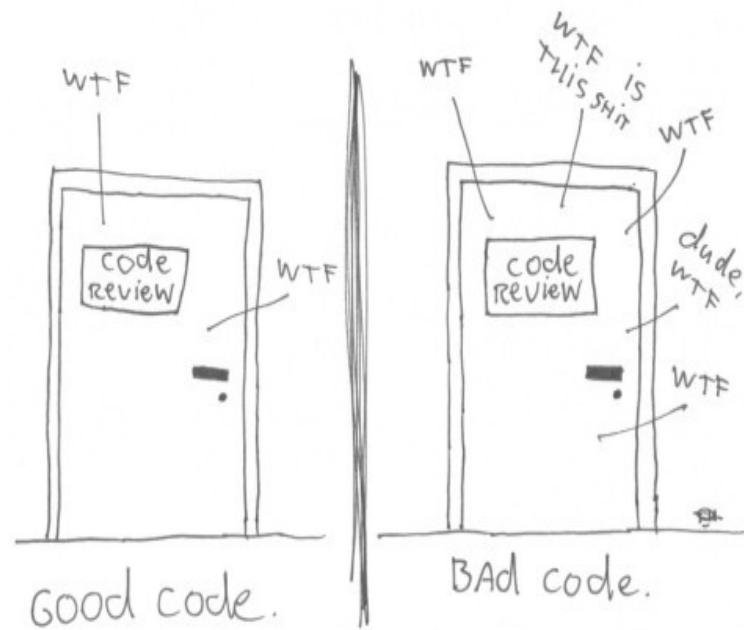
- ◆ The essence of code review is providing an author timely feedback on his or her changes

<sup>1</sup>Source: [http://en.wikipedia.org/wiki/Code\\_review](http://en.wikipedia.org/wiki/Code_review)

# Code Review Ensures Code Quality

- ◆ Average defect detection rate:
  - ◆ 25% for unit testing
  - ◆ 35% for functional testing
  - ◆ 45% for integration testing
  - ◆ 55% for design review
  - ◆ 60% for code review

The ONLY VALID MEASUREMENT  
OF CODE QUALITY: WTFs/MINUTE



# Code Review Ensures Safety

```
public void deposit(Account account, double amount) {  
    If (account.number == myAccountNumber) {  
        account.balance = account.balance + amount * 100;  
    } else {  
        account.balance = account.balance + amount;  
    }  
    .....  
}
```



# Code Review Improves Coding Skills

---



中新网  
Chinanews.com

# Types of Code Review

---

- ◆ Formal
  - ◆ Fagan Inspection (traditional)
- ◆ Lightweight

## Over-The-Shoulder

- A developer looks over the author's shoulder as the latter walks through the code

## E-mail Pass-Around

- Source code management system e-mails code to reviewers automatically after check-in is made

## Pair Programming

- Two authors develop code together at the same workstation

## Tool Assisted

- Authors and reviewers use specialized tools designed for peer code review

# Code Reviews in Industry

- ◆ Code reviews are a **very** common industry practice
- ◆ Made easier by advanced tools that:
  - ◆ integrate with configuration management systems
  - ◆ highlight changes (i.e., diff function)
  - ◆ allow traversing back into history
    - ◆ e.g., Eclipse, SVN tools

The screenshot shows a code review interface with two side-by-side code editors. Both editors are titled 'display.c 1.154'. The left editor shows the original code, and the right editor shows the modified code with red highlights indicating changes.

**Left Editor (Original Code):**

```
unsigned char *data;  
  
meta_error_trap_push_with_return (display);  
if (XGetWindowProperty (display->xdisplay,  
    event->xselectionrequest.requestor,  
    event->xselectionrequest.property, 0, 256, F  
    display->atom_atom_pair,  
    &type, &format, &n, &rest, &data) != Success  
{  
    meta_error_trap_pop_with_return (display, TRUE);  
    return;  
}  
  
if (meta_error_trap_pop_with_return (display, TRUE) == Success)  
{  
    /* FIXME: to be 100% correct, should deal with rest > 0,  
     * but since we have 4 possible targets, we will hardly ever  
     * meet multiple requests with a length > 8  
     */  
    adata = (Atom*)data;  
    i = 0;  
    while (i < (int) n)  
    {  
        if (!convert_property (display, screen,  
            event->xselectionrequest.requestor,  
            event->xselectionrequest.property,  
            adata[i], adata[i+1]))  
            adata[i+1] = None;  
        i += 2;  
    }  
}
```

**Right Editor (Modified Code):**

```
unsigned long num, rest;  
unsigned char *data;  
  
meta_error_trap_push (display);  
XGetWindowProperty (display->xdisplay,  
    event->xselectionrequest.requestor,  
    event->xselectionrequest.property, 0, 256, F  
    display->atom_atom_pair,  
    &type, &format, &n, &rest, &data);  
  
if (meta_error_trap_pop (display) == Success)  
{  
    /* FIXME: to be 100% correct, should deal with rest > 0,  
     * but since we have 4 possible targets, we will hardly ever  
     * meet multiple requests with a length > 8  
     */  
    adata = (Atom*)data;  
    i = 0;  
    while (i < (int) num)  
    {  
        if (!convert_property (display, screen,  
            event->xselectionrequest.requestor,  
            event->xselectionrequest.property,  
            adata[i], adata[i+1]))  
            adata[i+1] = None;  
        i += 2;  
    }  
}
```

# Checklist

---

- ◆ Does the code build correctly?
- ◆ Does the code execute as expected?
- ◆ Is all new code tested? Altered code tests up to date?
- ◆ Is the code readable?
- ◆ Do you understand the code you are reviewing?
- ◆ Has the developer tested the code?  
coding conventions?  
Are all functions, methods and classes documented?  
Are complex algorithms and code optimizations adequately commented?
- Error Handling
- Resource Leaks
- Thread Safeness
- Is the code free of unintended infinite loops?
- Are function parameters explicitly verified in the code?
- Pending/TODO
  - ◆ assertions
  - ◆ abnormal terminations
  - ◆ Database Transactions
  - ◆ Correct synchronization
  - ◆ no deadlocks/livelocks
  - ◆ Bug Fix Side Effects
  - ◆ All the occurrences of the bug are fixed



# Exercise

"Code review" this checkin.  
What feedback would you give the author?  
What changes would you demand before checkin?

```
public class Account {  
    double principal,rate;    int daysActive,accountType;  
  
    public static final int STANDARD=0, BUDGET=1,  
        PREMIUM=2, PREMIUM_PLUS=3;  
}  
...  
public static double calculateFee(Account[] accounts)  
{  
    double totalFee = 0.0;  
    Account account;  
    for (int i=0;i<accounts.length;i++) {  
        account=accounts[i];  
        if ( account.accountType == Account.PREMIUM ||  
            account.accountType == Account.PREMIUM_PLUS )  
            totalFee += .0125 * (           // 1.25% broker's fee  
                account.principal * Math.pow(account.rate*  
                    (account.daysActive/365.25))  
                - account.principal);          // interest-principal  
    }  
    return totalFee;  
}
```

# Improved code I

---

```
// blah blah blah
public class Account {
    private double principal;
    private double rate;
    private int daysActive;
    private Type type;

    // the various kinds of accounts our bank offers
    public enum Type {STANDARD, BUDGET, PREMIUM, PREMIUM_PLUS}

    // blah blah blah
    public double interest() {
        double years = daysActive / 365.25;
        double compoundInterest = principal * Math.pow(rate *
years);
        return compoundInterest - principal;
    }

    // blah blah blah
    public boolean isPremium() {
        return accountType == Type.PREMIUM ||
               accountType == Type.PREMIUM_PLUS;
    }
}
```

# Improved code 2

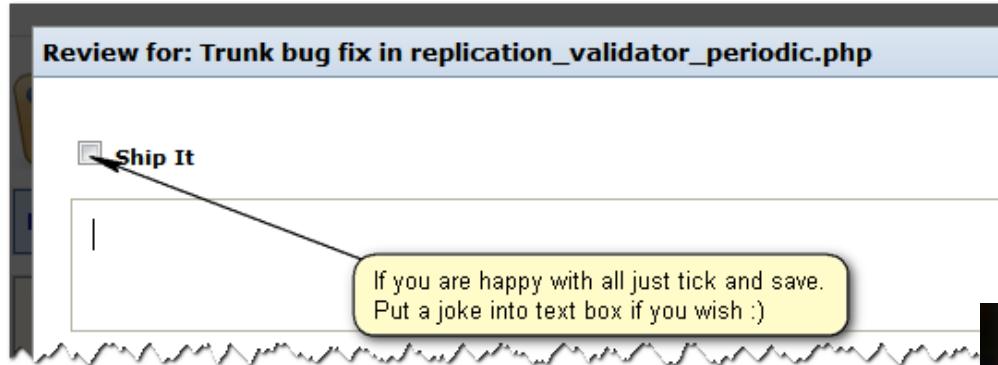
---

```
// blah blah blah
public static double calculateFee(Account accounts[]) {
    double totalFee = 0.0;
    for (Account account : accounts) {
        if (account.isPremium()) {
            totalFee += BROKER_FEE_PERCENT * account.interest();
        }
    }
    return totalFee;
}

// blah blah blah
public static final double BROKER_FEE_PERCENT = 0.0125;
}
```

# Ship It!

---



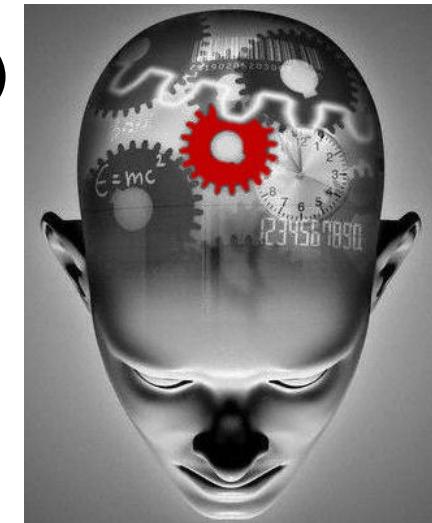
# Demo

---

# Tips for Effective Code Review

---

- ◆ Psychology – Let's face it, there's a lot of psychology in it, so play nice
- ◆ Be professional, nothing is personal
- ◆ Strive to understand, not criticize
- ◆ Get prepared
  - ◆ Understand the technology in use
  - ◆ Know the coding conventions
  - ◆ Read related code (classes/method being used)
- ◆ Read slow
- ◆ Short



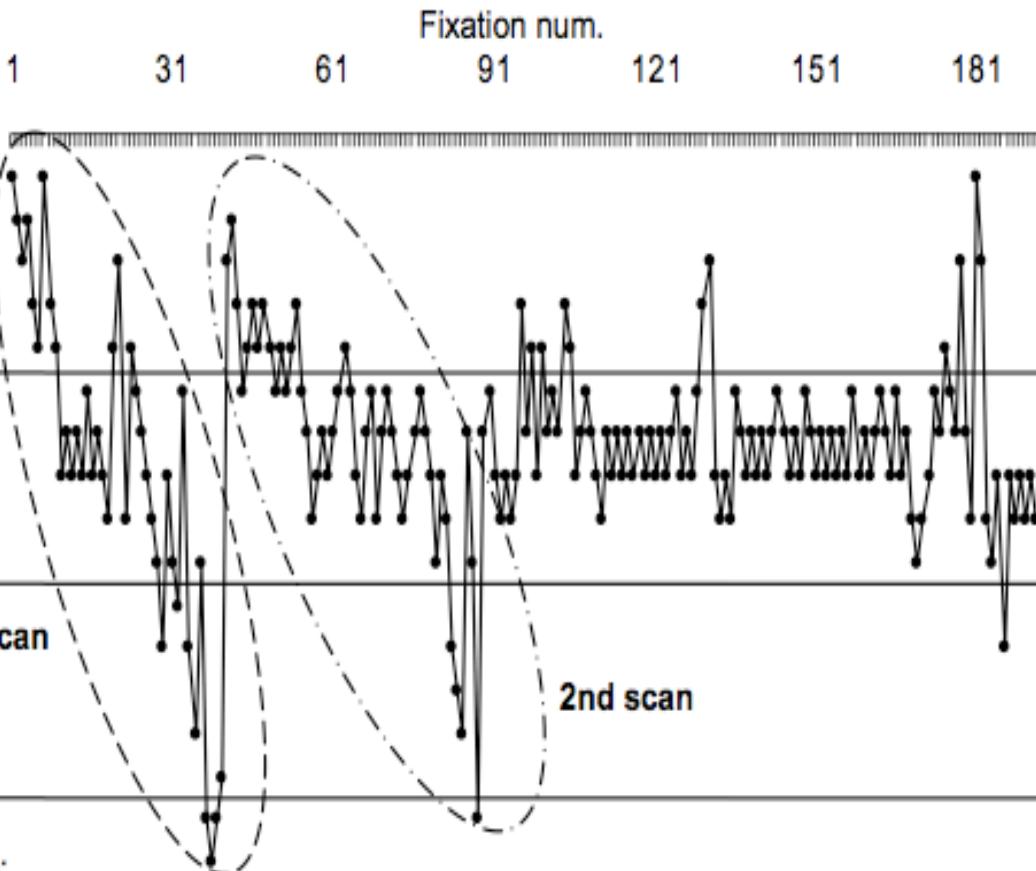
# Eye Tracking

---

```
01 void main(void) {  
02     int i, num, isPrime = 0;  
03  
04     printf("Input Number:");  
05     scanf("%d", &num);  
06  
07     i = 2;  
08     while(i < num) {  
09         if (num%i == 0)  
10             isPrime = 1;  
11         i = i + 1;  
12     }  
13  
14     if (isPrime == 1)  
15         printf("%d is prime number.\n", num);  
16     else  
17         printf("%d is NOT prime number.\n", num);  
18 }
```

# Eye Tracking

```
01 void main(void){  
02     int i, num, isPrime = 0;  
03  
04     printf("Input Number:");  
05     scanf("%d", &num);  
06  
07     i = 2;  
08     while(i < num){  
09         if(num%i == 0)  
10             isPrime = 1;  
11         i = i + 1;  
12     }  
13  
14     if(isPrime == 1)  
15         printf("%d is prime number.\n", num);  
16     else  
17         printf("%d is NOT prime number.\n", num);  
18 }
```

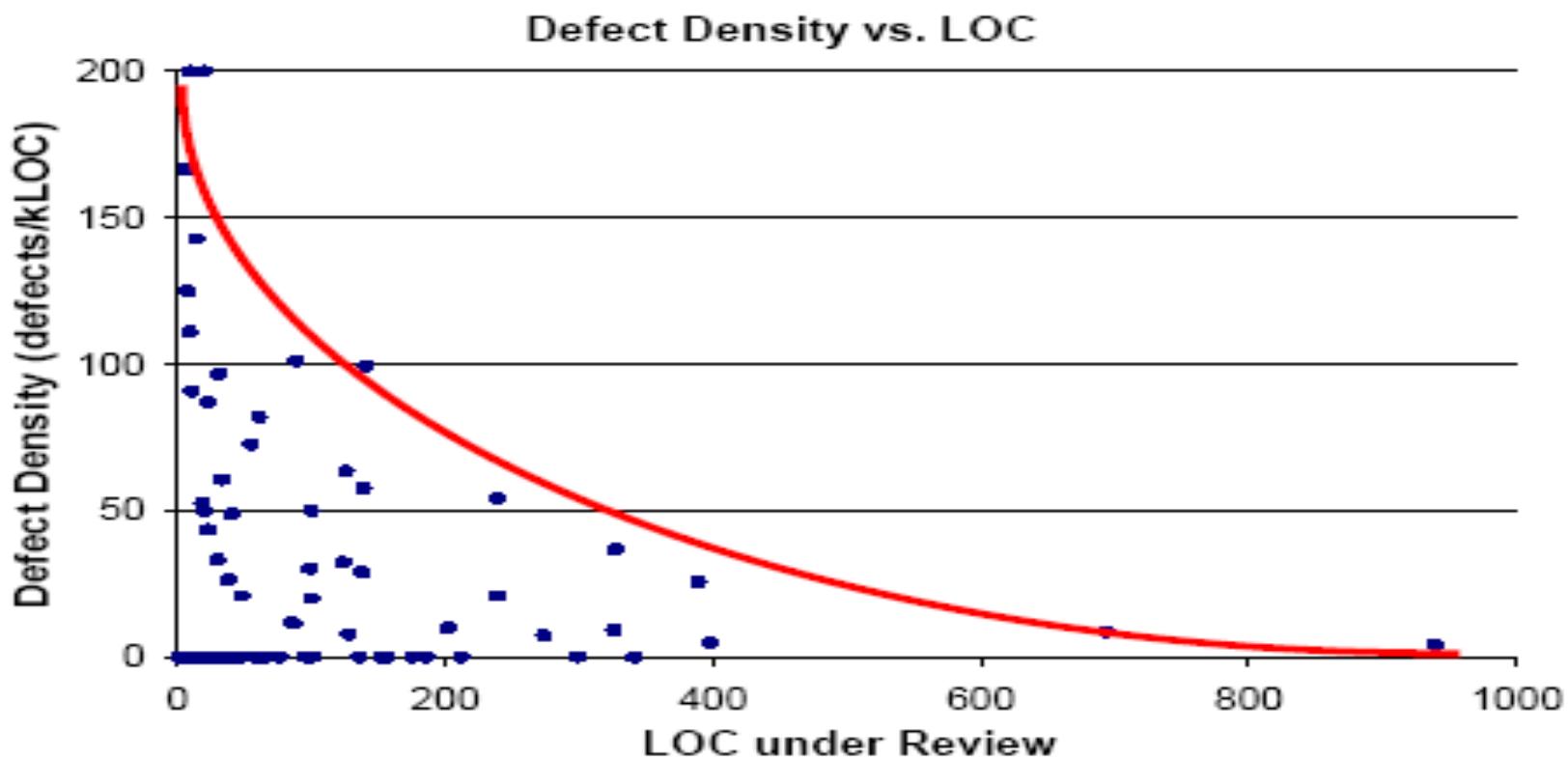


# Eye Tracking

---

- ◆ Now there's a physiological reason to write short method
- ◆ People who spend more time on the first scan find more issues and find them faster

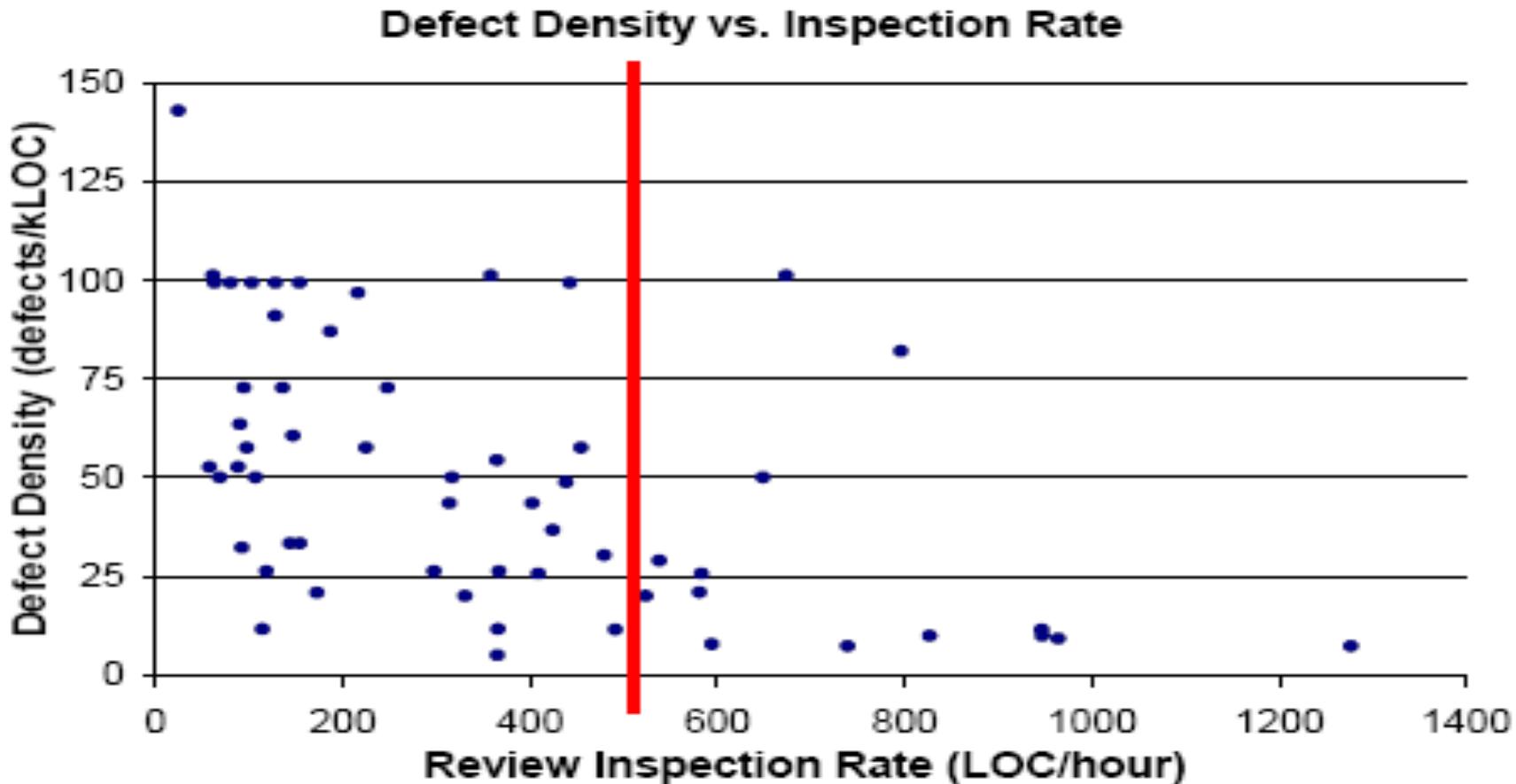
# Defect density vs LOC



Conclusion: Don't review too much code at once (<200-400 LOC)

<http://smartbear.com/white-paper.php?content=docs/articles/Case-Study.html>

# Defect density vs LOC/hour

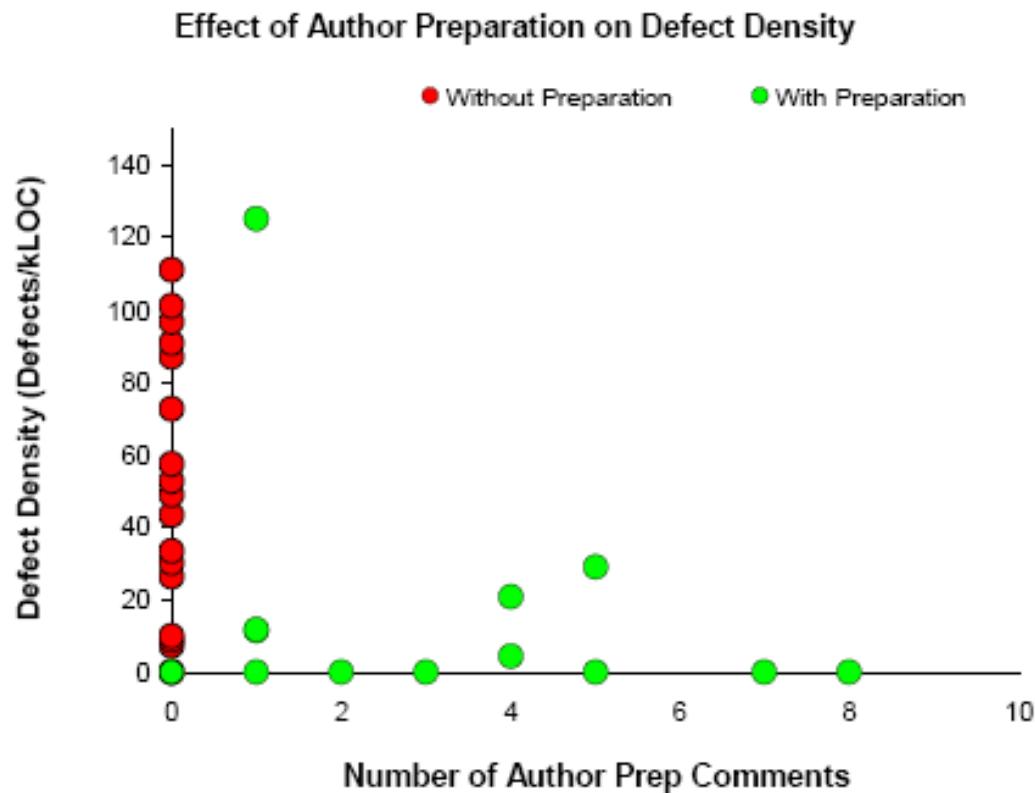


**Conclusion: Take your time (<500 LOC/hour)**

<http://smartbear.com/white-paper.php?content=docs/articles/CASE-STUDY.html>

# Author Preparation

---



**Conclusion:** Author preparation results in more efficient reviews  
<http://smartbear.com/white-paper.php?content=docs/articles/Case-Study.html>

# Code Review Exercise

---

- ◆ <http://cr.cs580.yusun.io/>
- ◆ Register an account if you haven't done it (with email)
  
- ◆ Send a CR request with your A4 code
- ◆ Ask 3 of your peers to review it
- ◆ cc CS580 group
  
- ◆ You should response to each of the comment
- ◆ You should upload new code for the change requests
  
- ◆ You should review others' code carefully
- ◆ You decide “Ship It” for others