Part I [Overview]
3. Quality Control

SE-307 Software Testing Techniques

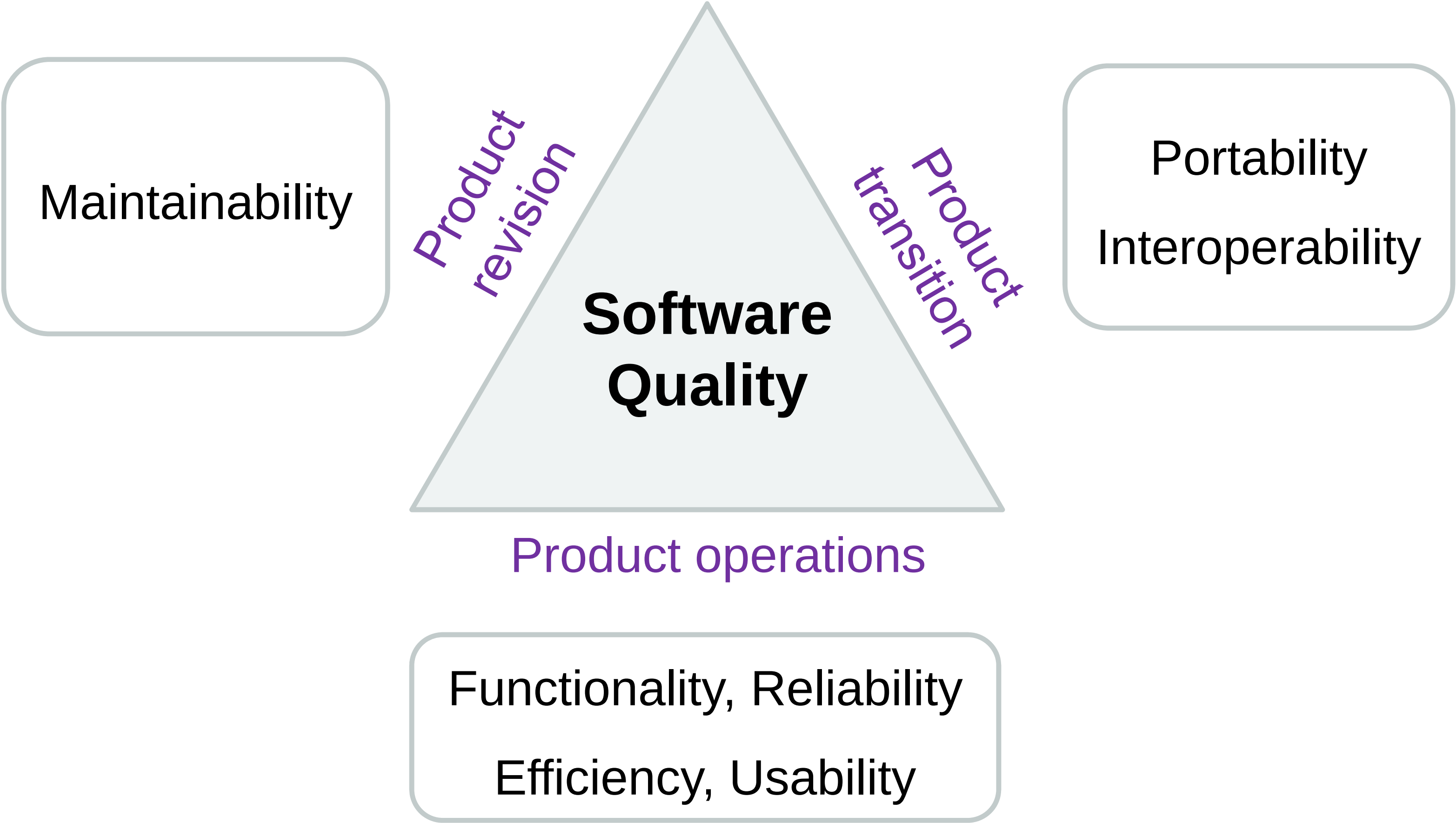
http://my.ss.sysu.edu.cn/wiki/display/SE307/Home

Instructor: Dr. Wang Xinming, School of Software, Sun Yat-Sen University

Review

- What we have learnt in the last lecture?
    - What is *software quality*?
    - What is *software bug*?

Software Quality Factors



Maintainability

Product revision

Software Quality

Product transition

Portability

Interoperability

Product operations

Functionality, Reliability

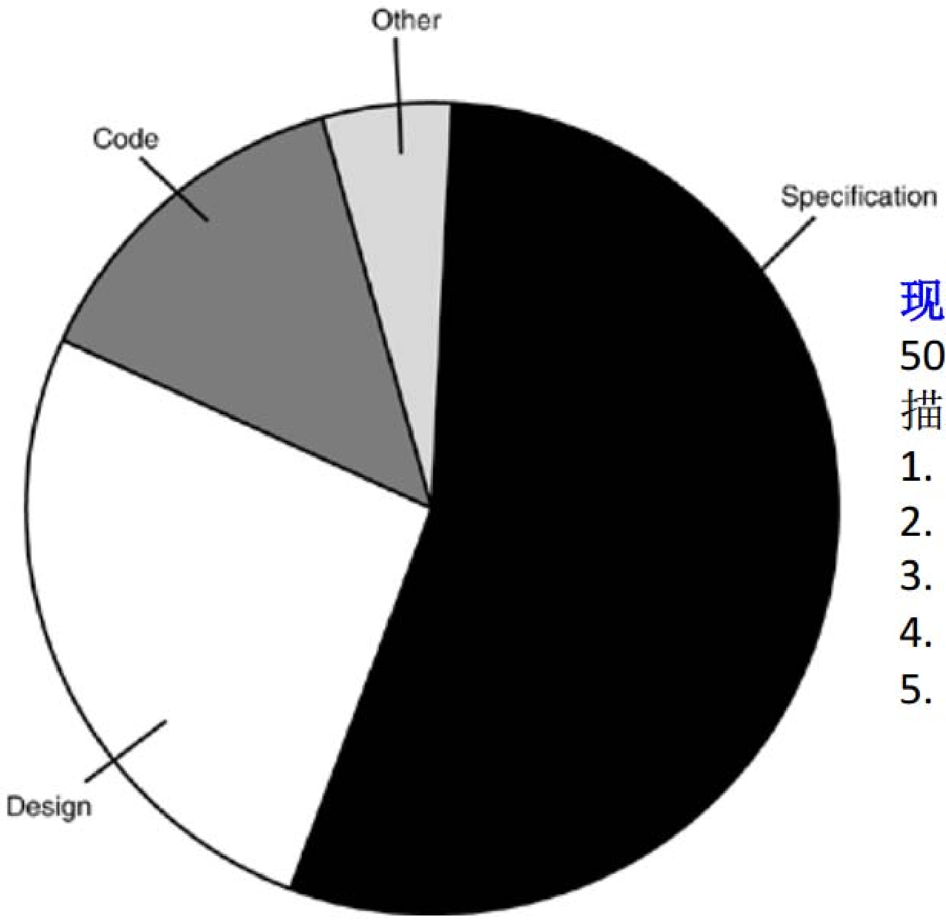
Efficiency, Usability

Some additional facts you shall know

- Bugs are much more than coding mistakes

代码问题<30%。
主要原因：

1. 代码测试技术
2. 编程习惯
3. COTS集成
4. ……



现实的项目调查：
50%以上来源与需求描述。主要原因：

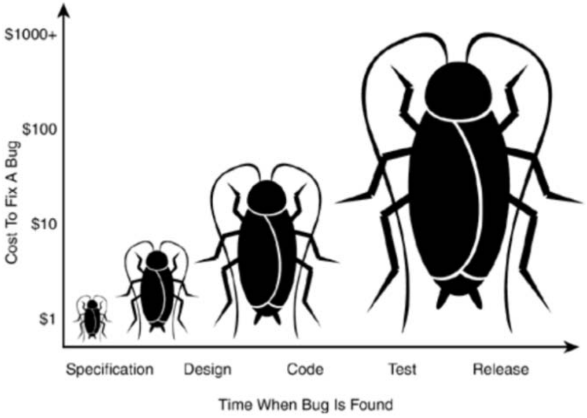
1. 需求描述不全面
2. 需求随时间变化
3. 需求描述有歧义
4. 用户沟通失败
5. ……

比例变化因素：

1. 规范化程度
2. 应用领域
3. 重视程度
4. ……

Some additional facts you shall know

- The earlier a bug is found, the cheaper it costs.
  - The Y2K (Year 2000) Bug
    - In the early 1970s, The computer had very limited memory for storage, forcing to conserve every last byte.
    - One method was to shorten dates from their 4-digit format, such as 1973, to a 2-digit format, such as 73.
    - The problems occur when the current year hit 2000.
  - Results
    - There are millions of software using short date format world wide.
    - It's estimated that several hundred billion dollars were spent, worldwide, to update computer programs, to fix potential Year 2000 failures



Some additional facts you shall know

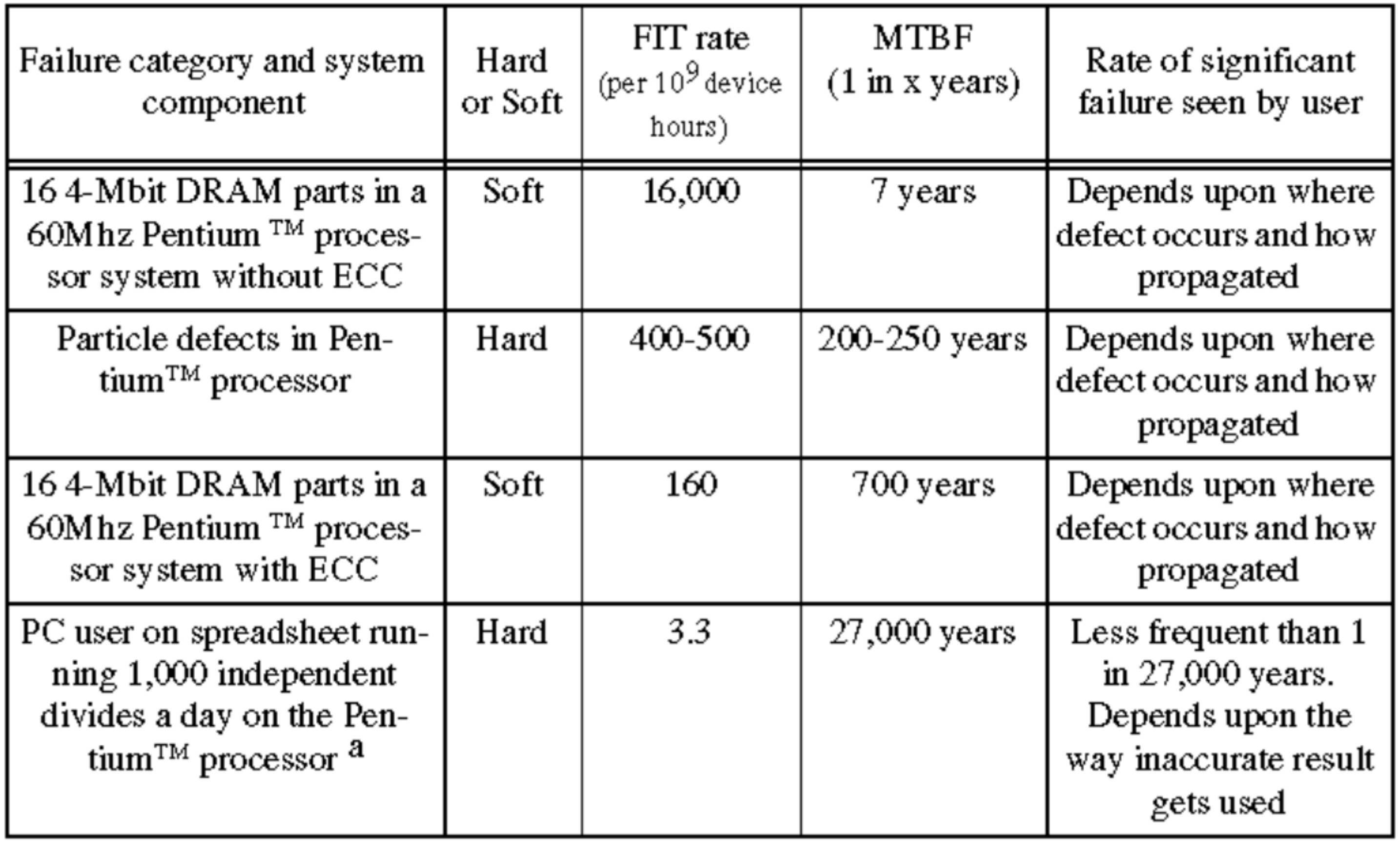
- Not all software bugs get fixed after they are exposed.
  - Intel Pentium Floating-Point Division Bug, 1994
    - Bug in the SRT radix-4 division algorithm.
    - On October 30, 1994, Dr. Thomas find it on his Pentium PC. He posted his find on the Internet.
  - Results
    - Their software test engineers had found the problem while performing their own tests before the chip was released.
    - Intel's management decided that the problem wasn't severe enough or likely enough to warrant fixing it or even publicizing it.
    - In the end, Intel took a charge of more than $500 million to cover the costs of replacing bad chips

$$\frac{4195835}{3145727} = 1.333820449136241002$$

$$\frac{4195835}{3145727} = 1.333739068902037589$$

Intel’s Analysis

Fail!



Failure category and system component	Hard or Soft	FIT rate (per 10^9 device hours)	MTBF (1 in x years)	Rate of significant failure seen by user
16 4-Mbit DRAM parts in a 60Mhz Pentium ™ processor system without ECC	Soft	16,000	7 years	Depends upon where defect occurs and how propagated
Particle defects in Pentium™ processor	Hard	400-500	200-250 years	Depends upon where defect occurs and how propagated
16 4-Mbit DRAM parts in a 60Mhz Pentium ™ processor system with ECC	Soft	160	700 years	Depends upon where defect occurs and how propagated
PC user on spreadsheet running 1,000 independent divides a day on the Pentium™ processor [a]	Hard	3.3	27,000 years	Less frequent than 1 in 27,000 years. Depends upon the way inaccurate result gets used

The exact meaning of bug: Failure, Fault, and Error

D'OH!
KA-BOOOM

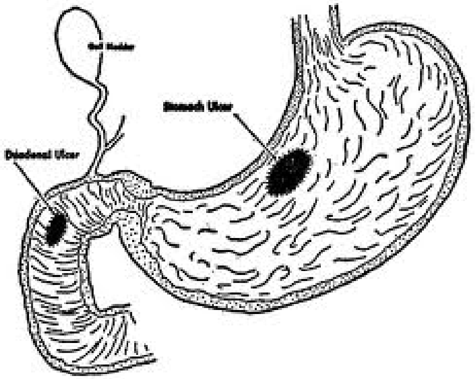
can lead to

can lead to

Error

Fault

Failure



Historical notes on bugs

- *It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise—this thing gives out and [it is] then that **'Bugs'**—as such little faults and difficulties are called—show themselves and months of intense watching, study and labor are requisite. . .*

  – Thomas Edison

- *An analyzing process must equally have been performed in order to furnish the Analytical Engine with the necessary operative data; and that herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, the cards may give it wrong orders.*

  –Ada, Countess Lovelace

- *Stemming from the first bug, today we call errors or glitch in a program a **bug**.*

  - Grace Hopper

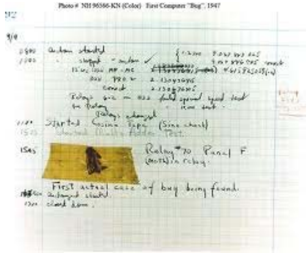


Photo # NH 96566-KN (Color) First Computer "Bug", 1947

9/9

0800 Anctan started

1000 " stopped - anctan ✓ [illegible]

Relays 6-2 in 033 failed special speed test

In Relay [illegible]

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

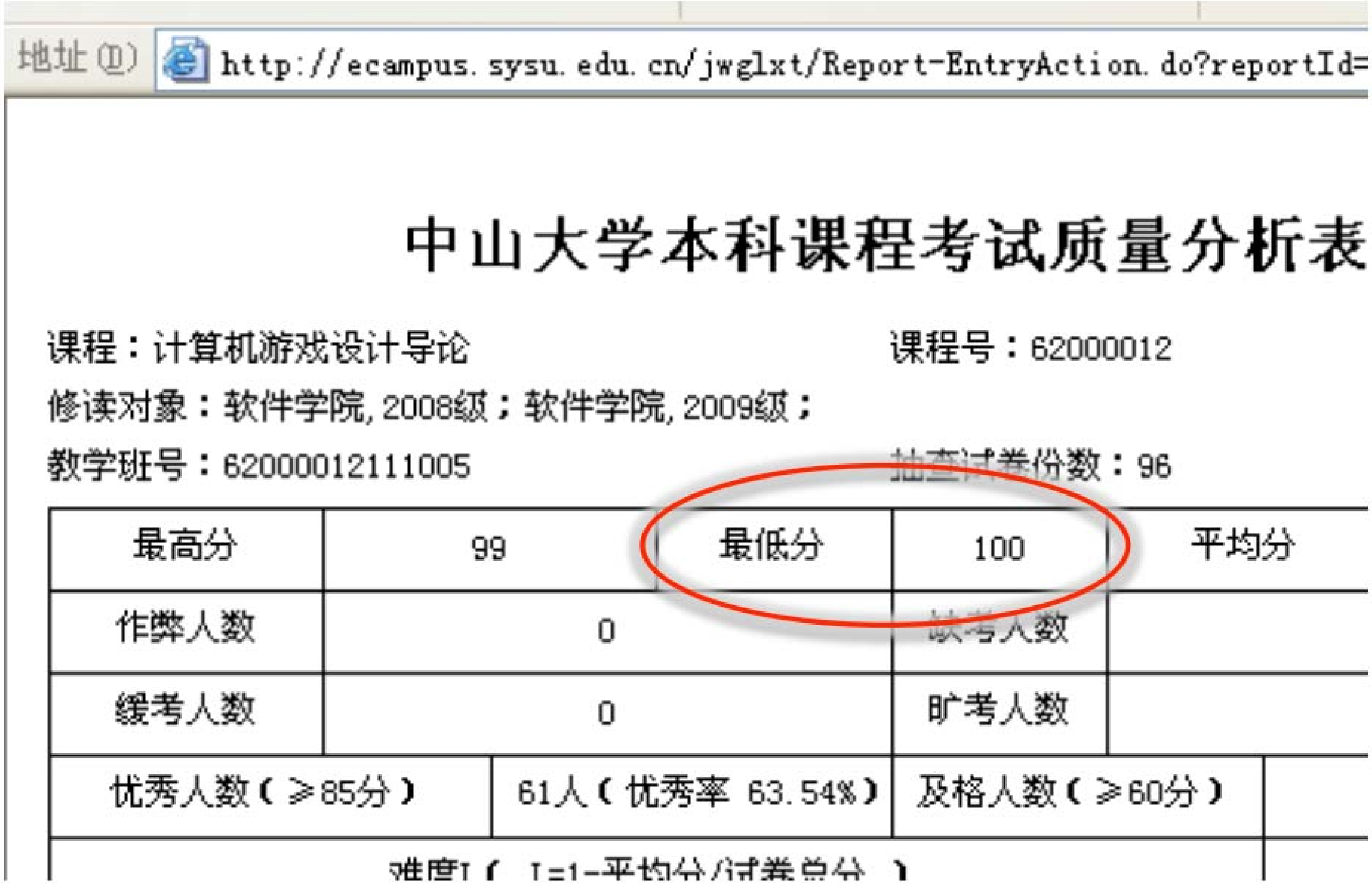
1630 Antangent started.

1700 closed down.

The high cost of bug

- NIST report, “The Economic Impacts of Inadequate Infrastructure for Software Testing” (2002)
  - Inadequate software testing costs the US alone between $22 and $59 billion annually.
  - Better approaches could cut this amount in half
- Huge losses due to web application failures
  - Financial services : $6.5 million per hour (just in USA!)
  - Credit card sales applications : $2.4 million per hour (in USA)
- In Dec 2006, amazon.com BOGO offer turned into a double discount

Alarm: bugs are among us!



http://ecampus.sysu.edu.cn/jwglxt/Report-EntryAction.do?reportId=

中山大学本科课程考试质量分析表

课程：计算机游戏设计导论　　课程号：62000012

修读对象：软件学院,2008级；软件学院,2009级；

教学班号：62000012111005　　抽查试卷份数：96

最高分	99	最低分	100	平均分
作弊人数	0	缺考人数		
缓考人数	0	旷考人数		
优秀人数（≥85分）	61人（优秀率 63.54%）	及格人数（≥60分）		
难度I（ I=1-平均分/试卷总分 ）				

Questions last time

- Share your hairiest bug war stories.

CACM, 1997, Vol. 40, No. 4

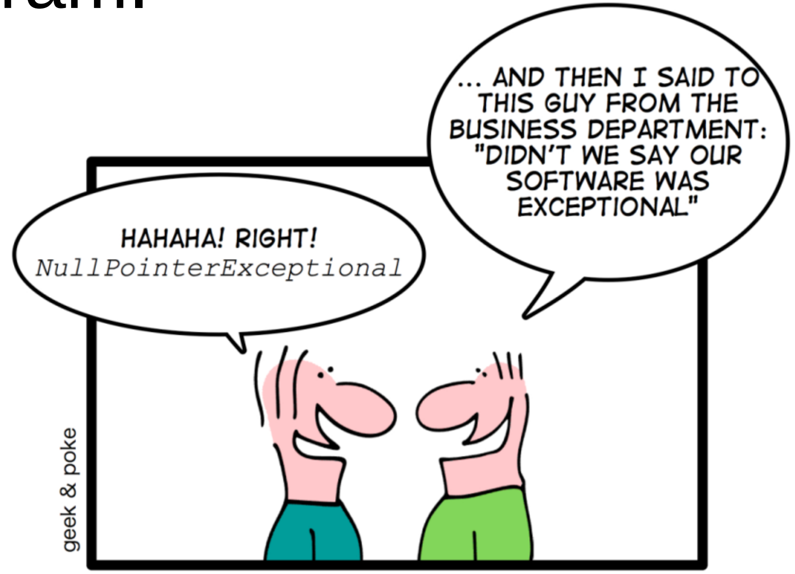
Marc Eisenstadt

My Hairiest Bug War Stories

Despite the availability of industrial-strength debuggers and integrated program development environments, professional programmers still engage in far more detective work than they should have to. This is their story—along with some of the ways they might eradicate their bugs.

STUDIES OF THE PSYCHOLOGY OF COMPUTER PROGRAMming and debugging [1, 2, 5, 8–12], while important in their own right, have generally overlooked the potential benefit of self-reports by programmers that reflect the phenomenology of debugging, that is, what it's really like out there from the programmer's perspective. However, two exceptions to this observation are the detailed account by Knuth of using a logbook to document all the errors he encountered over a 10-year development period working on TeX [6] and a logbook of the development efforts of a team implementing the [illegible]

- Share your opinion on why it is so difficult to write bug-free program.


... AND THEN I SAID TO THIS GUY FROM THE BUSINESS DEPARTMENT: "DIDN'T WE SAY OUR SOFTWARE WAS EXCEPTIONAL"
HAHAHA! RIGHT! NullPointerExceptional
geek & poke

Difficulty of writing bug-free programs

Complexity

- Many parameters
- No two parts of software are alike
- System goes through many states during execution
- Inherent non-linearity

Conformity

- System should conform with standards imposed by
  - other components, such as hardware, or
  - external bodies, such as existing software

Changeability

- Constant need for change

Bottom-line

- People do make mistakes when they build complex systems
  - no matter how good their intentions are,
  - no matter how good their technology is,
  - no matter how high their skill levels are.

OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
11/13 © 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTERNOON!

Outline

- What is *Software Quality Control* and *Software Quality Assurance*?
- What is *verification* and *validation*?
- Techniques for verification.

Quality Control & Quality Assurance

- Quality Assurance （质量保障）：
  - *A set of activities designed to ensure that the development and/or maintenance process is adequate to ensure a system will meet its objectives.*
- Quality Control （质量控制）：
  - *A set of activities designed to evaluate a developed work product.*
- Differences:
  - QA focus on the process elements of a project. QC activities focus on finding defects in specific deliverables.
  - QA makes sure you are doing the right things, the right way, QC makes sure the results of what you've done are what you expected.
  - QA is ***process oriented*** and QC is ***product oriented***.

- Question: is testing QC or QA?

Further differences between QC & QA

- In CMMI, QA is independent of the project team.
  - Observe, but not participate, the development process.
  - Might come from clients.
  - Report to authority higher than PM.
  - Interact with PM, QC, and authority higher than PM.

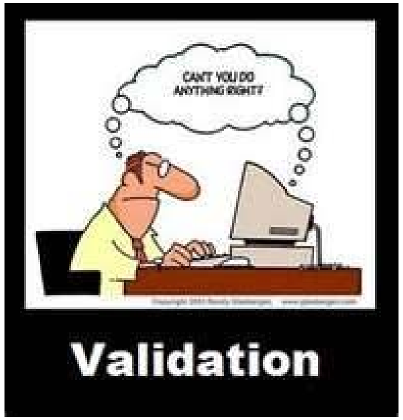
- I hate QA people! (*And they hate you too*) .
  - “*They whine about everything they're sent. I think we should all boycott QA for a week, send them nothing, and put them out of a job.*”
  - Like QA or not, you need to work with them.

Further differences between QC & QA

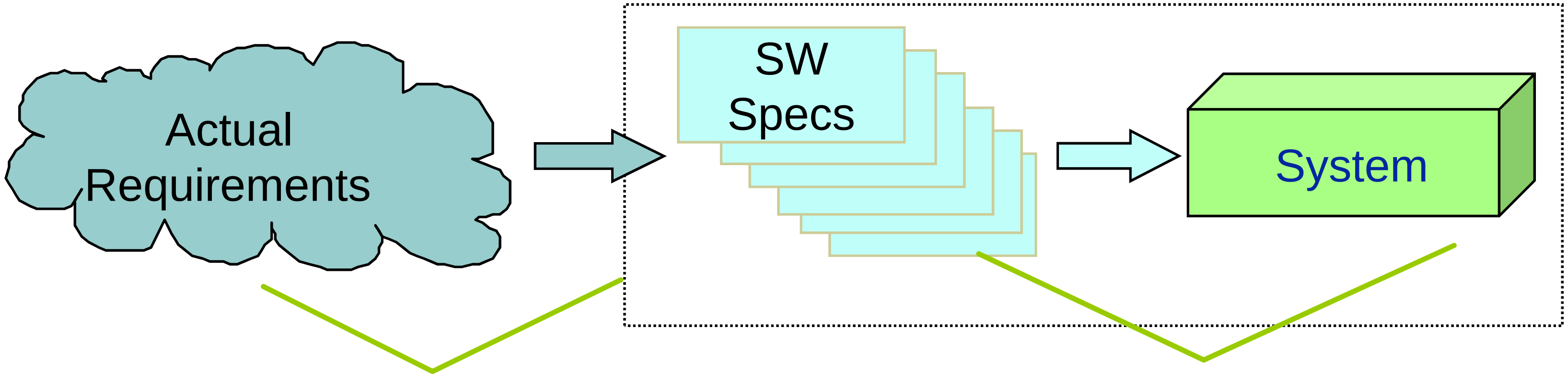
- QC is part of the project team.
  - Examine whether the product satisfies the quality requirement.
  - When a bug is found, fix it or find a developer to fix it.
  - Last gate between the product and the customers.
  - Not all developers can do a good QC.
  - A good QC requires deep technology knowledge.
- Common perception: “Bad programmers do QC”
  - Completely wrong.
  - Google：I am QC, you are QC, everyone in the team is QC.
- Two kind of QC: *verification* and *validation*

Verification and Validation

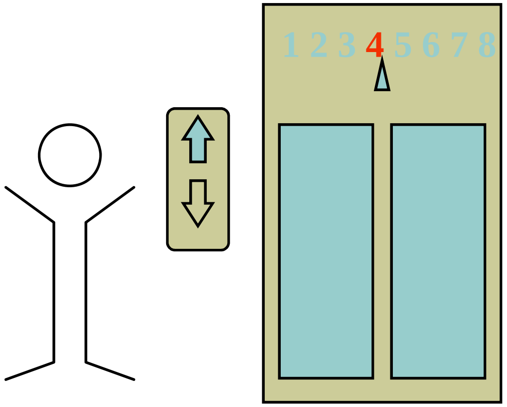
- **Validation**:
  does the software system meets the user's real needs?
  are we building the right software?



- **Verification**:
  does the software system meets the requirements specifications?
  are we building the software right?



Verification or validation?

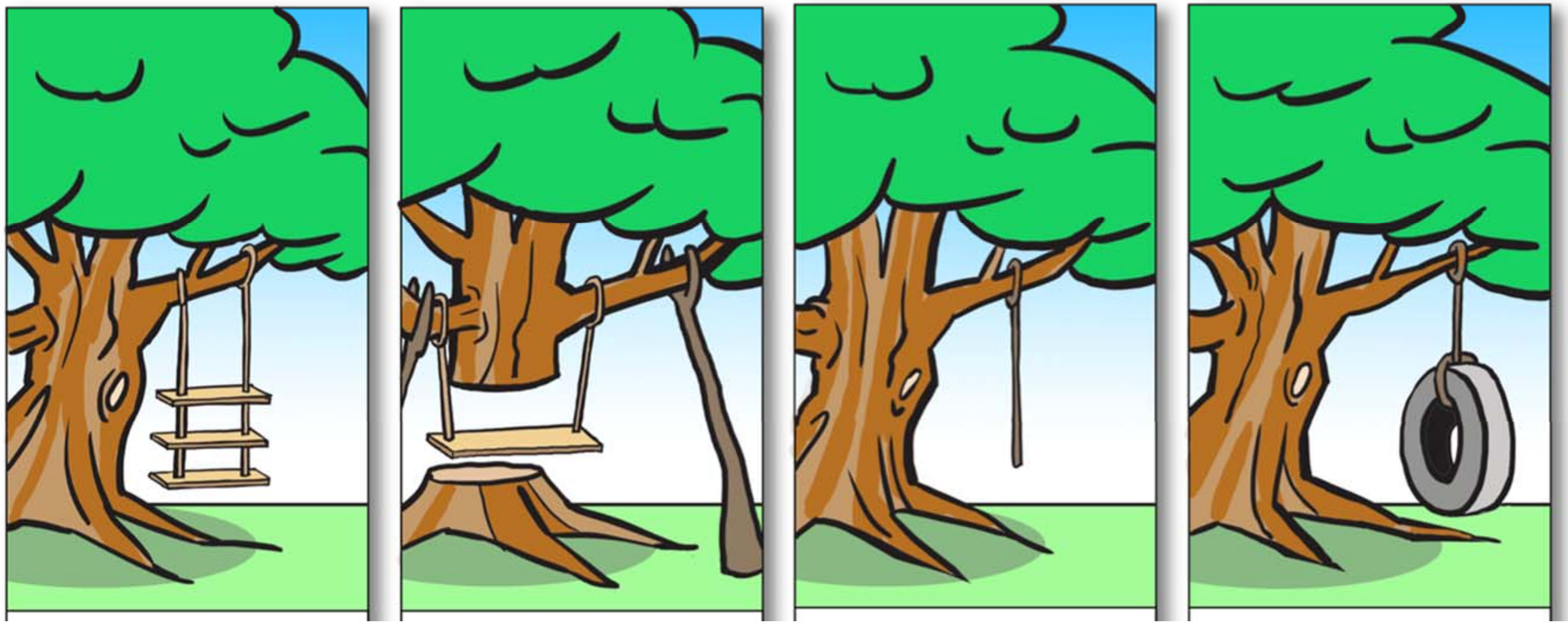


Example: elevator response

Unverifiable (but validatable) spec: ... if a user presses a request button at floor i, an available elevator must arrive at floor i soon...

Verifiable spec: ... if a user presses a request button at floor i, an available elevator must arrive at floor i within 30 seconds...

Verification or Validation?



1. The requirement gathered from customers.

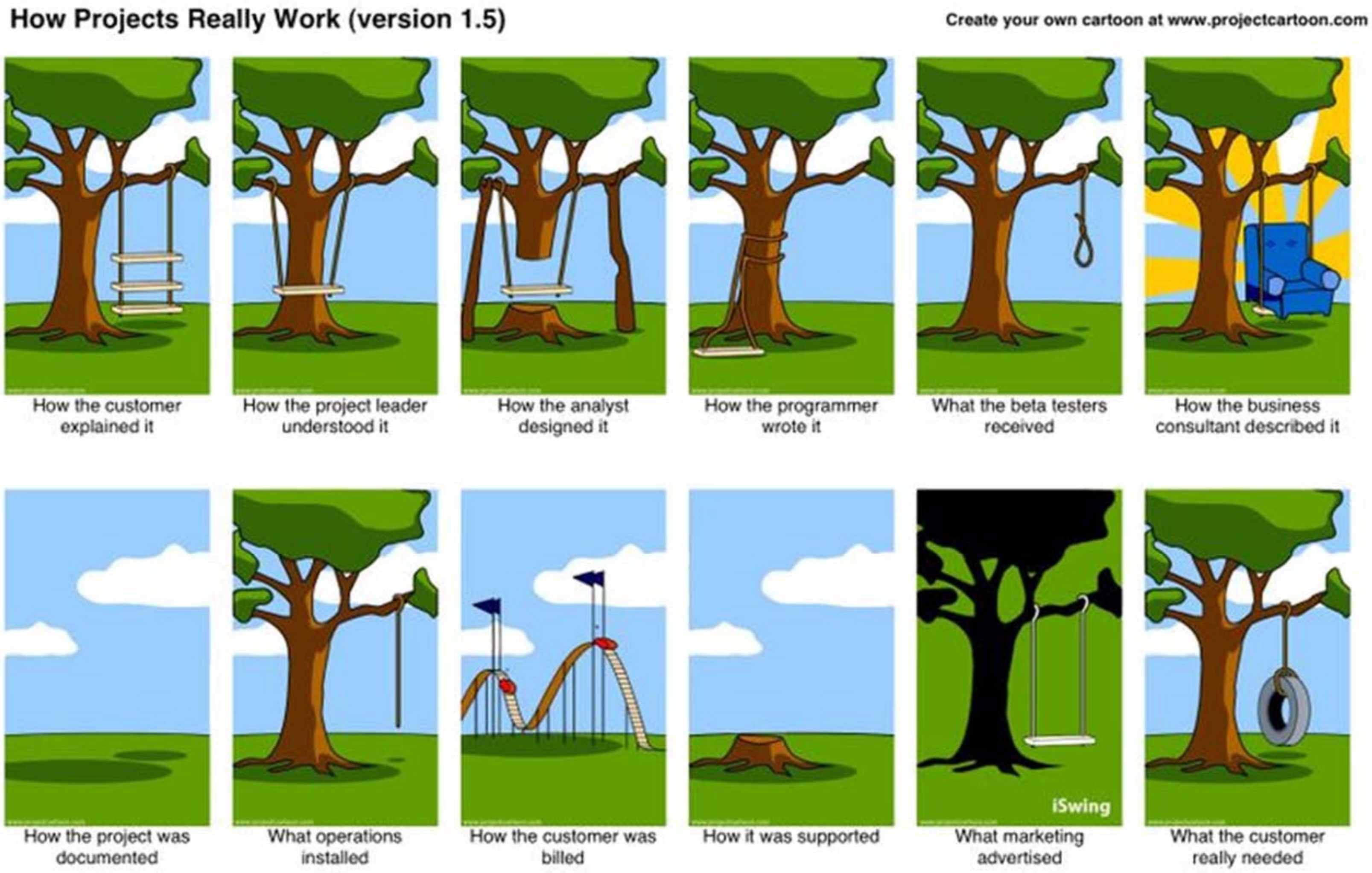
2. The design of the system.

3. The implementation

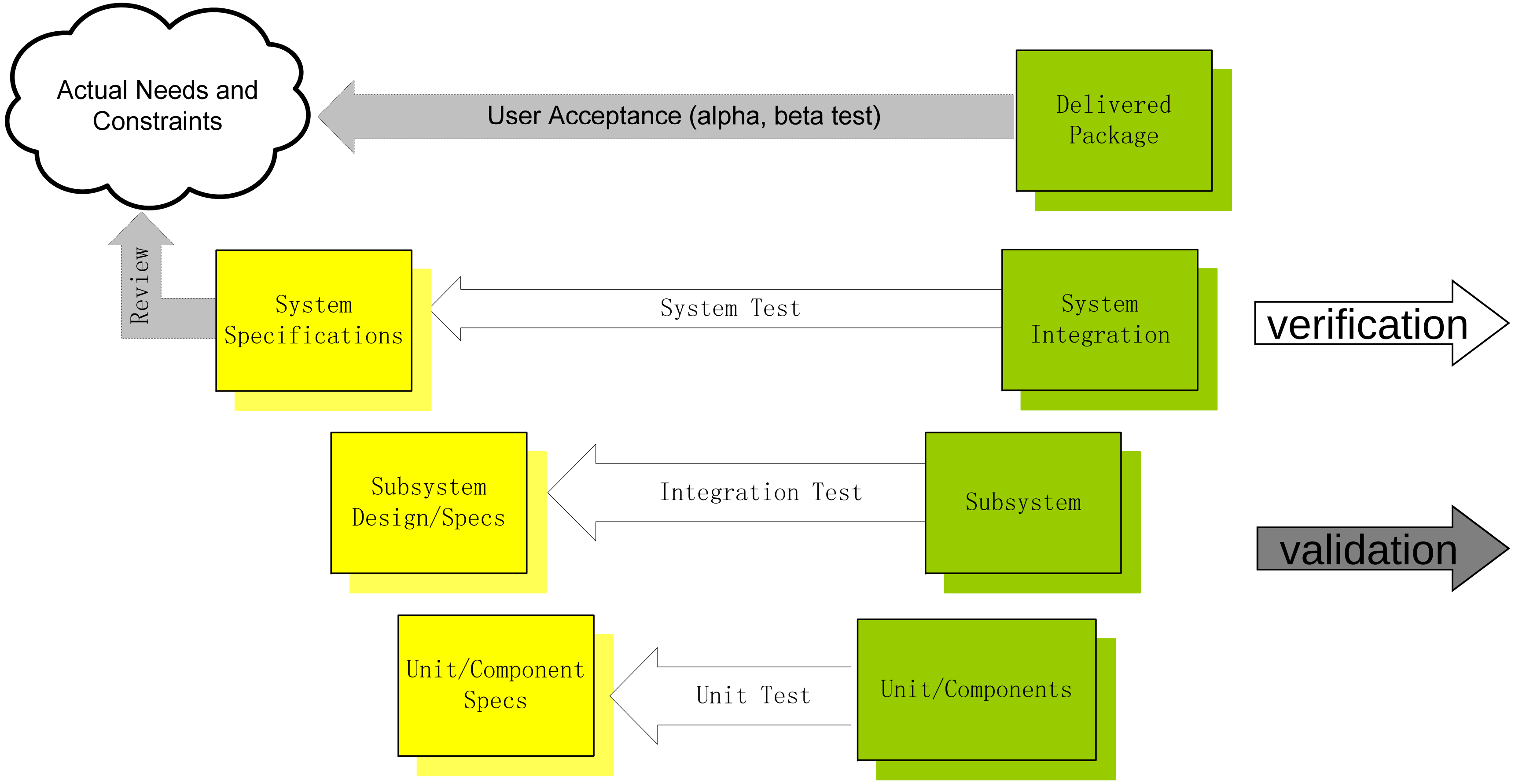
4. What the customer really wants.

Verification or Validation?

威力加强版：


How Projects Really Work (version 1.5)
Create your own cartoon at www.projectcartoon.com
How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

Validation and Verification Activities



Software Verification Techniques

- **Inspection**
  - A constructive review of the artifacts, typically the code (code review)
- **Model checking**
  - Proving that the programs in question to be absolutely correct with respect to some formal properties on *any* input
- **Program analysis**
  - Identifying certain problematic code patterns that are known to be bug-prone.
- **Testing**
  - Finding bugs by executing input cases to a program and comparing the outputs with some reference system.

Sound vs. Complete Static vs. Dynamic

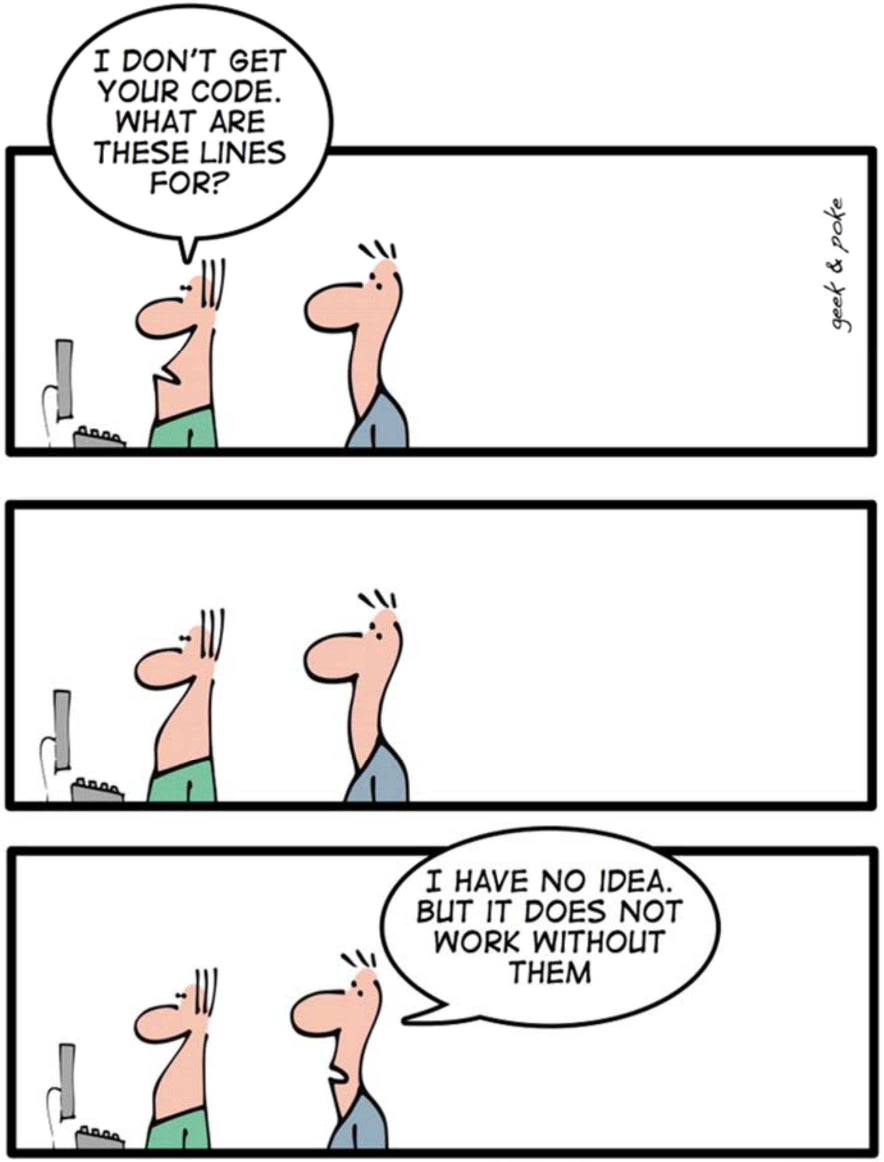
- The characteristic of a technique:
  - **Sound**: An analysis of a program P with respect to a property p is sound if the analysis returns true only when the program does satisfy this property.
  - **Complete**: An analysis of a program P with respect to a property p is complete if the analysis always returns true when the program actually does satisfy the property
  - **Static:** An analysis is static if it can infer the properties without running the program.
  - **Dynamic**: An analysis is dynamic if it can infer the properties by running the program.

Inspection (code reviews)

- ***who***: Original developer and reviewer, sometimes together in person, sometimes offline.
- ***what***: Reviewer gives suggestions for improvement on a logical and/or structural level, to conform to previously agreed upon set of quality standards.
  - Feedback leads to refactoring, followed by a 2nd code review.
  - Eventually reviewer approves code.
- ***when***: When code author has finished a coherent system change that is otherwise ready for checking
  - change shouldn't be too large or too small
  - *before* committing the code to the repository or incorporating it into the new build

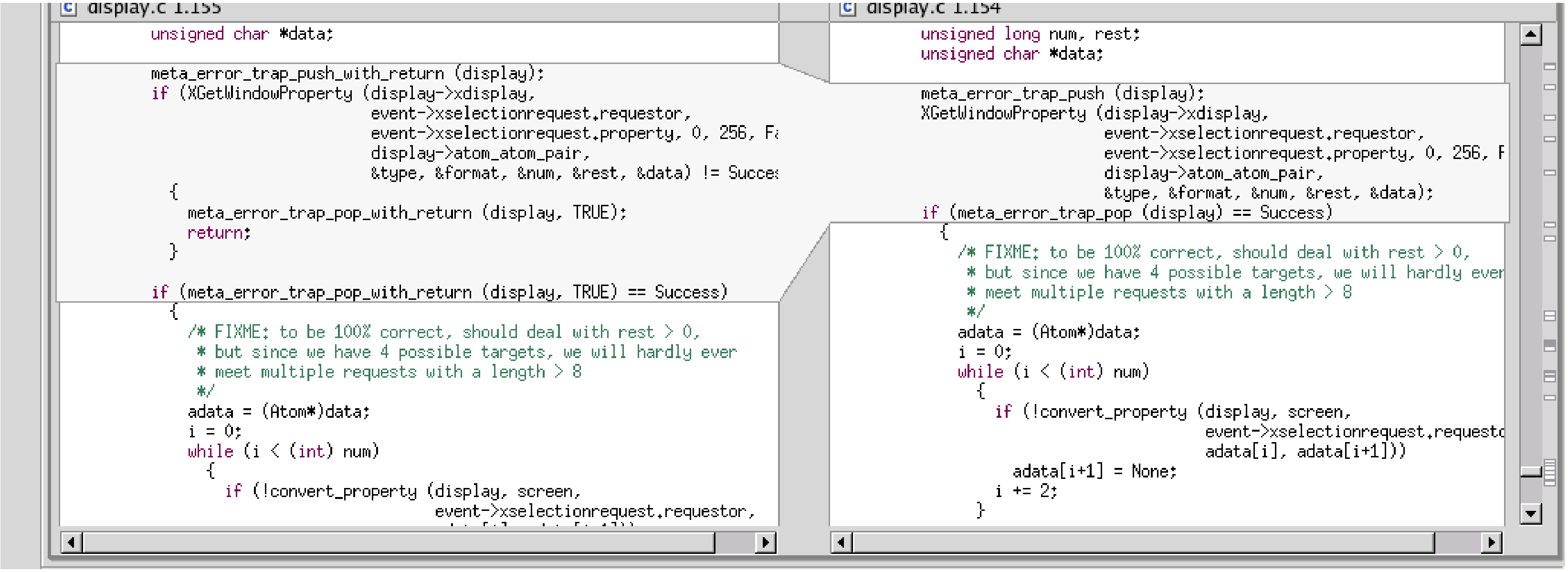
The merit of Inspection

- Two heads are better than one.
- Force the programmer to keep it simple and stupid (KISS)
- Any piece of code is understood by at least two personals.


I DON'T GET YOUR CODE. WHAT ARE THESE LINES FOR?
geek & poke
I HAVE NO IDEA. BUT IT DOES NOT WORK WITHOUT THEM

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



display.c 1.155

```
        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, &num, &rest, &data) != Succe
          {
            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) num)
              {
                if (!convert_property (display, screen,
                                       event->xselectionrequest.requestor,
```

display.c 1.154

```
        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, &num, &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.requesto
                                       adata[i], adata[i+1]))
                  adata[i+1] = None;
                i += 2;
              }
```

Example 1: Google

- “All code that gets submitted needs to be reviewed by at least one other person, and either the code writer or the reviewer needs to have readability in that language. Most people use Rietveld* to do code reviews, and obviously, we spend a good chunk of our time reviewing code."

-- Amanda Camp, Software Engineer, Google

Rietveld: https://developers.google.com/appengine/articles/rietveld

Example 2: Facebook

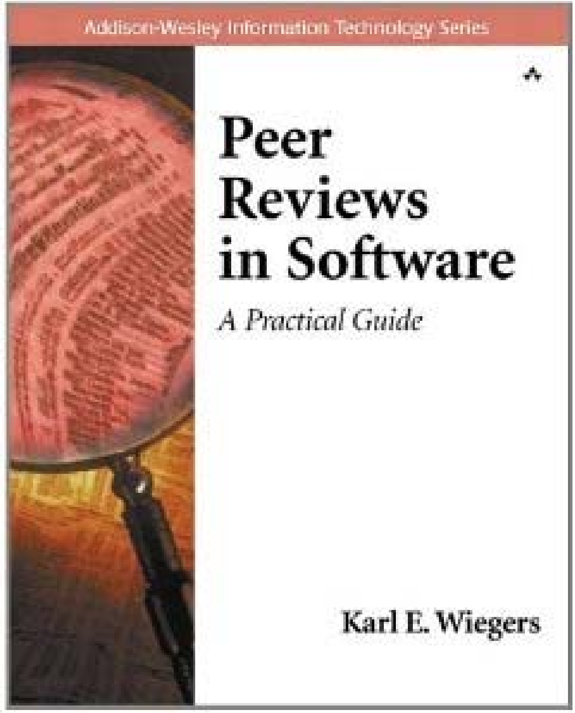
- "At Facebook, we have an internally-developed web-based tool to aid the code review process. Once an engineer has prepared a change, she submits it to this tool, which will notify the person or people she has asked to review the change, along with others that may be interested in the change -- such as people who have worked on a function that got changed.

  At this point, the reviewers can make comments, ask questions, request changes, or accept the changes. If changes are requested, the submitter must submit a new version of the change to be reviewed. All versions submitted are retained, so reviewers can compare the change to the original, or just changes from the last version they reviewed. Once a change has been submitted, the engineer can merge her change into the main source tree for deployment to the site during the next weekly push, or earlier if the change warrants quicker release."

- Ryan McElroy, Software Engineer, Facebook

Pros and Cons

- The good:
  - You write better code when you know it will be reviewed.
  - A second (or thirc ; will help spot defects.
  - More than one pe ece of code.
  - Expose problems clarity, and lack of conformity.
- The limitation:
  - Can be casual.
  - Can be destructiv
  - Alone is insufficie .
    - Tend to catch obvious, “stupid” bugs.



Addison-Wesley Information Technology Series

Peer Reviews in Software

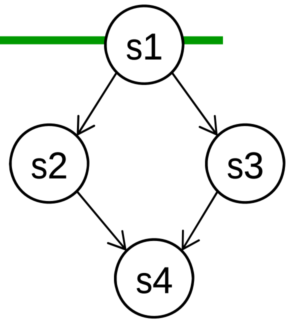
A Practical Guide

Karl E. Wiegers

Model Checking

- Calculate whether a system satisfies a certain behavioral property:
  - Is the system deadlock free?
  - Whenever a packet is sent will it eventually be received?
- So it is like testing? No, major difference:
  - Look at *all* possible behaviors of a system
- Automatic, if the system is finite-state
  - Potential for being a push-button technology
  - Almost no expert knowledge required
- How do we describe the system?
- How do we express the properties?

Model Checking



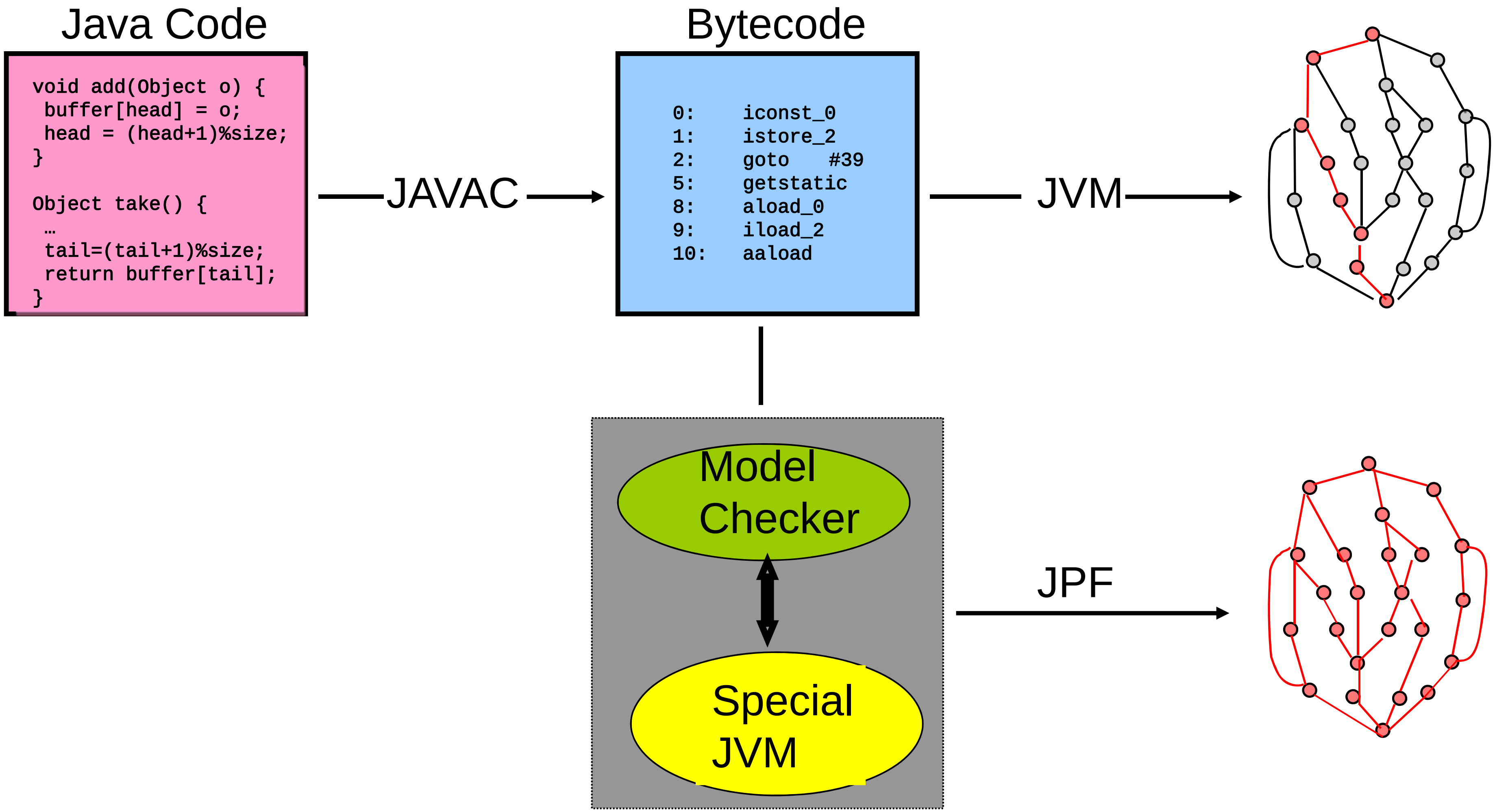
- User requirement => *formal properties*
  - x>y at line 100
  - i>=j+1 as loop invariant
  - Array bound is never exceeded
  - If a request is sent, acknowledgement is eventually received
- Explore feasible paths and symbolic states of the program
  - Static
  - Complete but not sound

Program

```
int x;
int y=8,z=0,w=0;
Input(x);
if (x)
  z = y – 1; //bug: z=y+1
else
  w = y + 1;
assert (z == 7 || w == 9)
```

Example 1: JPF

http://javapathfinder.sourceforge.net/



Java Code

```
void add(Object o) {
 buffer[head] = o;
 head = (head+1)%size;
}

Object take() {
 …
 tail=(tail+1)%size;
 return buffer[tail];
}
```

JAVAC

Bytecode

```
0:    iconst_0
1:    istore_2
2:    goto    #39
5:    getstatic
8:    aload_0
9:    iload_2
10:   aaload
```

JVM

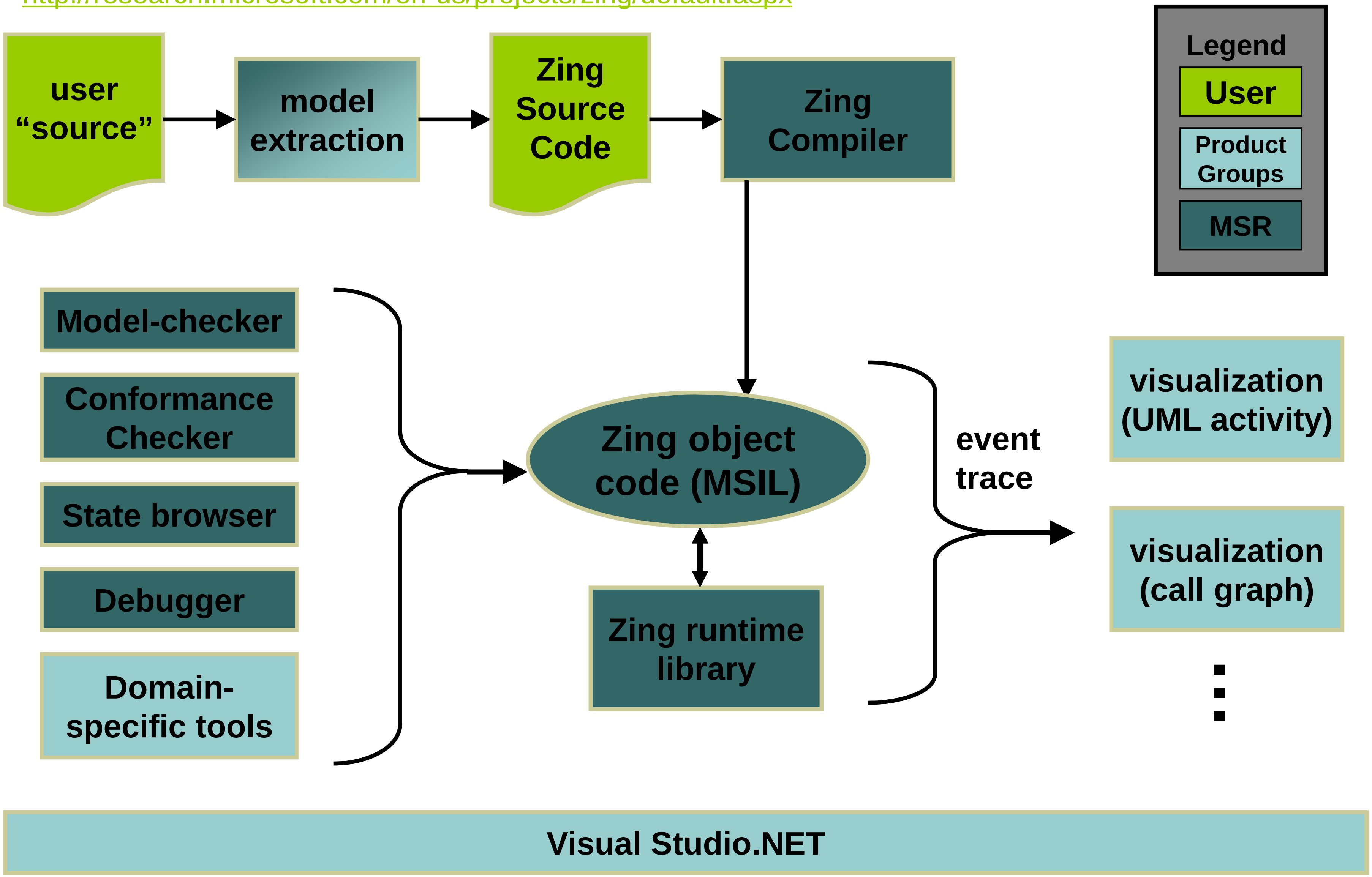
Model Checker

Special JVM

JPF

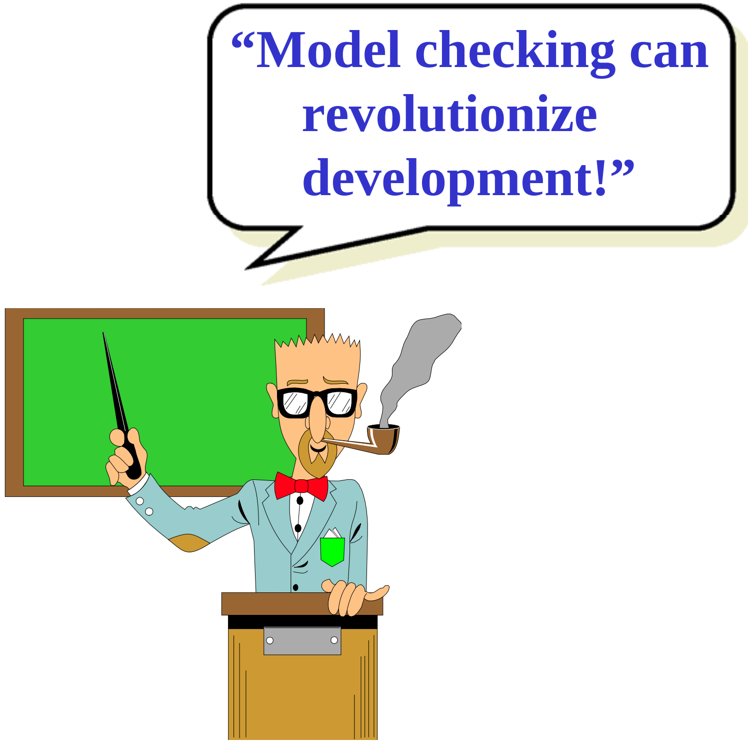
Example 2: Microsoft Zing

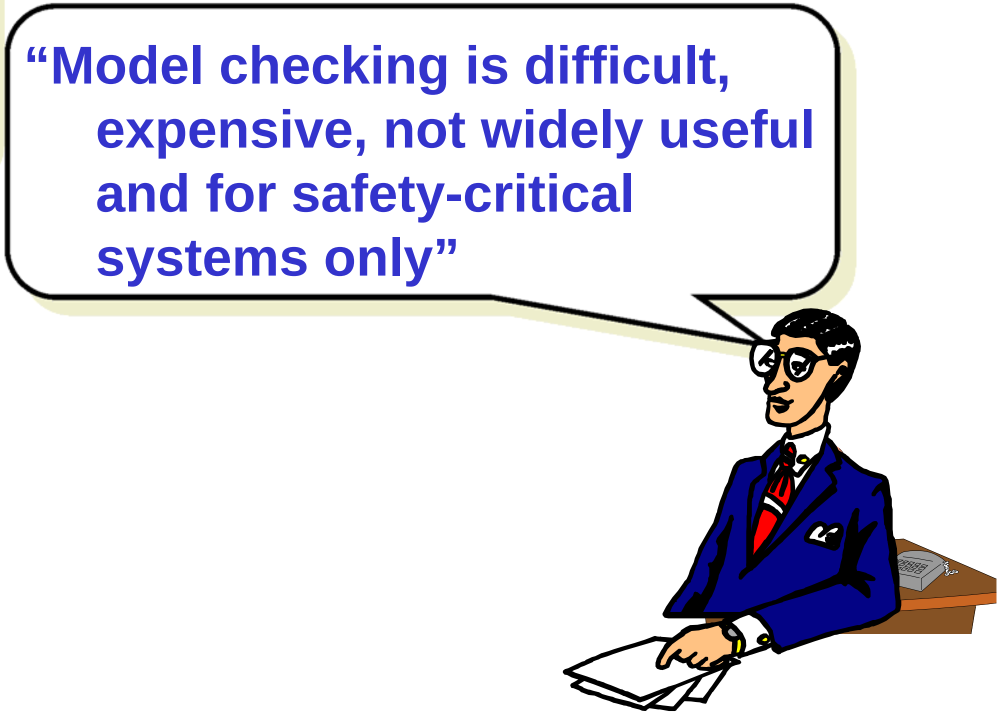
http://research.microsoft.com/en-us/projects/zing/default.aspx



Pros and Cons

- The good
  - Proof of no-bug
- The bad
  - Expensive




“Model checking is difficult, expensive, not widely useful and for safety-critical systems only”

Program analysis

- Target certain known bug-prone code patterns
  - Type analysis
  - Dead code analysis
  - Escape analysis
  - Clone analysis
  - Pointer analysis
- Match the program against these patterns
  - Static
  - Complete, some are sound

Program

```
int x;
int y=8,z,w;
input(x);
if (x)
  z = y – 1; //bug: z=z-1;
else
  w = y + 1;
```

Program analysis in the industry

- Microsoft
  - Monthly central runs of global analysis tools
    - PREfix (Inter-procedural symbolic evaluation), ESP (Inter-procedural path-sensitive dataflow analysis)
    - Defects auto-inserted into central bug database
    - Ranking, filtering, triage, support
    - Release management drives the bugs
  - Developer desktop use of local analysis tools
    - PREfast, espX (Intra-procedural abstract interpretation)
    - Installed and enabled by default for all developers
    - Tools run incrementally with the build
  - Both sets of tools report around 3000 bugs every month

Example: FindBug

http://findbugs.sourceforge.net/

- Concentrates on detecting potential bugs and performance issues
- Can detect many types of common, hard-to-find bugs

NullPointerException

```
Address address = client.getAddress();
if ((address != null) || (address.getPostCode() != null)) {
 ...
}
```

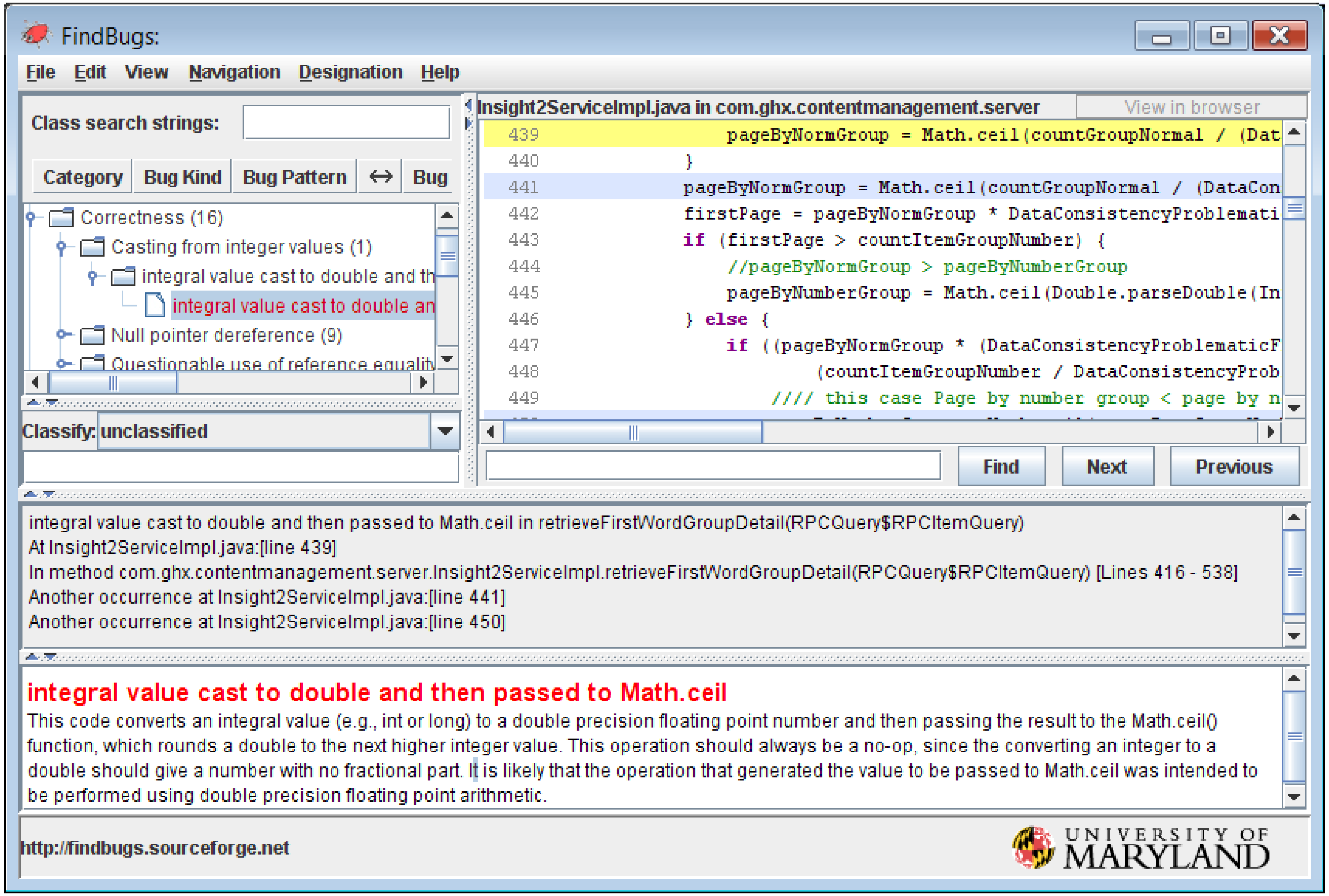
Uninitialized field

```
public class ShoppingCart {
    private List items;
    public addItem(Item item) {
        items.add(item);
    }
}
```

Example: FindBug

http://findbugs.sourceforge.net/

Can run as a standard tool or a Eclipse plugin


FindBugs:
File Edit View Navigation Designation Help
Class search strings:
Category | Bug Kind | Bug Pattern | ↔ | Bug
Correctness (16)
Casting from integer values (1)
integral value cast to double and th
integral value cast to double an
Null pointer dereference (9)
Questionable use of reference equality
Classify: unclassified
Insight2ServiceImpl.java in com.ghx.contentmanagement.server
View in browser
Find
Next
Previous
http://findbugs.sourceforge.net
UNIVERSITY OF MARYLAND

```
439             pageByNormGroup = Math.ceil(countGroupNormal / (Dat
440         }
441         pageByNormGroup = Math.ceil(countGroupNormal / (DataCon
442         firstPage = pageByNormGroup * DataConsistencyProblemati
443         if (firstPage > countItemGroupNumber) {
444             //pageByNormGroup > pageByNumberGroup
445             pageByNumberGroup = Math.ceil(Double.parseDouble(In
446         } else {
447             if ((pageByNormGroup * (DataConsistencyProblematicF
448                     (countItemGroupNumber / DataConsistencyProb
449                 //// this case Page by number group < page by n
```

integral value cast to double and then passed to Math.ceil in retrieveFirstWordGroupDetail(RPCQuery$RPCItemQuery)
At Insight2ServiceImpl.java:[line 439]
In method com.ghx.contentmanagement.server.Insight2ServiceImpl.retrieveFirstWordGroupDetail(RPCQuery$RPCItemQuery) [Lines 416 - 538]
Another occurrence at Insight2ServiceImpl.java:[line 441]
Another occurrence at Insight2ServiceImpl.java:[line 450]

integral value cast to double and then passed to Math.ceil

This code converts an integral value (e.g., int or long) to a double precision floating point number and then passing the result to the Math.ceil() function, which rounds a double to the next higher integer value. This operation should always be a no-op, since the converting an integer to a double should give a number with no fractional part. It is likely that the operation that generated the value to be passed to Math.ceil was intended to be performed using double precision floating point arithmetic.

Pros and Cons

- Pros
  - Quick
  - Cheap to apply
- Cons
  - False positive
  - Might mask the real causes

Testing

- Pick up a set of input data that satisfies certain adequacy requirement

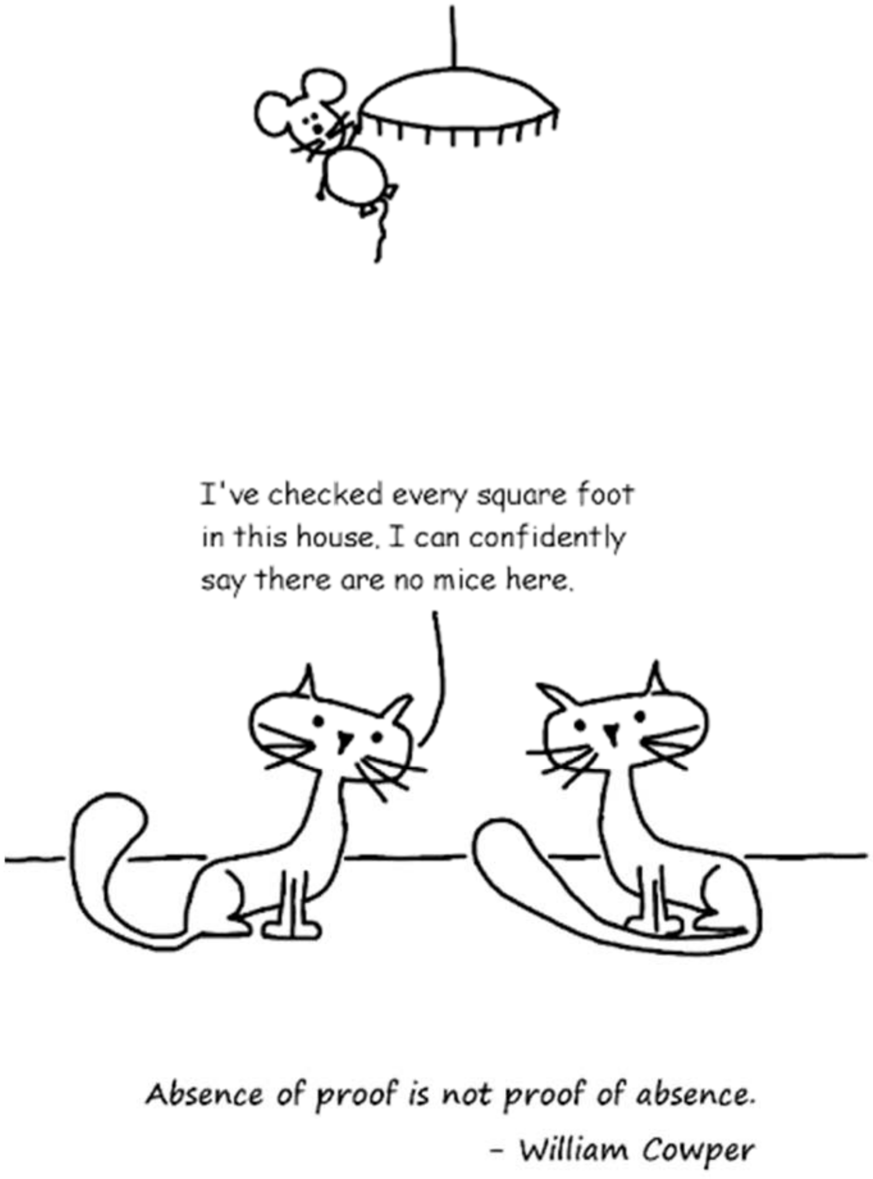
  e.g. cover all statements/paths, cover all boundary cases

- Run the program, compare the actual output with expected value.
  - Dynamic
  - Sound but not complete

Program

```
int x;
int y=8,z=0,w=0;
input(x);
if (x)
  z = y – 1; //bug: z=y+1
else
  w = y + 1;
output(z);
```

The limit of testing


I've checked every square foot in this house. I can confidently say there are no mice here.
Absence of proof is not proof of absence.
- William Cowper

Formal definitions of testing

- IEEE definition:
  - The process of exercising or evaluating a system or system components by manual or automated means to verify that it satisfies specific requirements and to identify difference between expected and actual results.
- *The art of software testing, Glenford J. Myers*
  - Testing is the process of executing a program with the intent of finding errors."
  - Successful (positive) test: exposes an error

What testing is not

- “Testing is the process of demonstrating that an errors are not present”
  - Testing can never show that an error is not present.
- “Testing is the process of establishing confidence that the program does what it intends to do”
  - Testing can never achieve this end.

Which QC techniques shall I choose?

- There are no fixed recipes.
- QC engineer must:
  - choose and schedule the right blend of techniques
    - to reach the required level of quality
    - within cost constraints
  - design a specific solution that suits
    - the problem
    - the requirements
    - the development environment

Example: Windows Driver Model SDK uses model checking to find potential IRQL_NOT_LESS_OR_EQUAL bugs

Review

- What is *Software Quality Control and Software Quality Assurance*?
- What is *verification* and *validation*?
- Techniques for verification.

Thank you!