中山大學
SUN YAT-SEN UNIVERSITY

# Part I [Overview]
# 2. Software Quality and Bugs


## 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**

# Outline

- What is *software quality*?
- What is *software bug*?

# What is *software quality*?

- What is the difference between a shanzai (山寨) phone and authentic (原厂) phone?




***Not a singular / primitive concept***

*different views*: product *vs.* user-based, …

*multi-dimensional*: many attributes of interest

*multi-level*: broad *vs.* specific, …

***No absolute right or wrong***

*context dependent*: constraints, compromises,…

# Some Professional Definitions

- "***Fitness for purpose***"

  (Juran and Gryna, 1970)
- " ***Zero defects*** – conformance to requirements"

  (Crosby, 1979)
- "The degree to which the attributes of the software enable it to ***perform its specified end item use***."

  (DoD, 1985)
- "The totality of features and characteristics of a product or service that bear on its ability to ***satisfy stated or implied needs***." (ISO, 1986)
- "Quality is when ***the customer comes back***, not the product."

  (Frühauf, 1994)

# Software Quality Factors

Maintainability

Product revision

Product transition

Portability

Interoperability

**Software Quality**

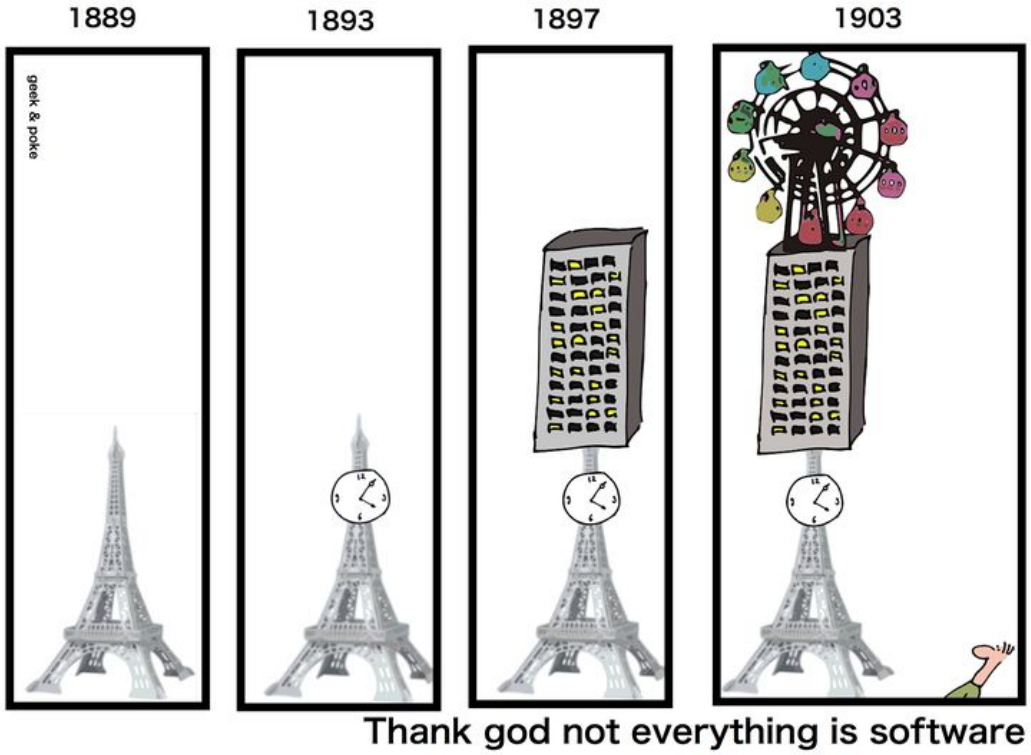
Product operations

Functionality, Reliability

Efficiency, Usability

# Functionality

- Give the user what they want in a correct way, and nothing more.

# Functionality

- **Suitability:** The capability of the software to provide an appropriate set of functions for specified tasks and user objectives.
- **Correctness:** The capability of the software to provide the right or agreed.
- **Compliance**: The capability of the software to adhere to application related standards, conventions or regulations in laws and similar prescriptions.
- **Security**: The capability of the software to prevent unintended access and resist deliberate attacks intended to gain unauthorised access to confidential information, or to make unauthorised modifications to information or to the program so as to provide the attacker with some advantage or so as to deny service to legitimate users.

# Functionality failure

## A correctness bug in Windows Calculator:

Proud to exist since Windows 3.1

Step 1: Open Calculator

Step 2: Click ‘4’

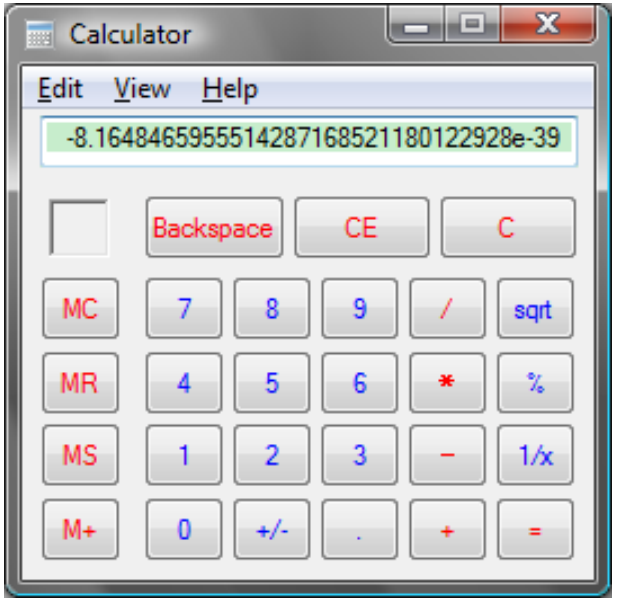
Step 3: Click ‘sqrt’

Step 4: Click ‘-’

Step 5: Click ‘2’

Step 6: Click ‘=‘

Shouldn’t the result be 0 !?

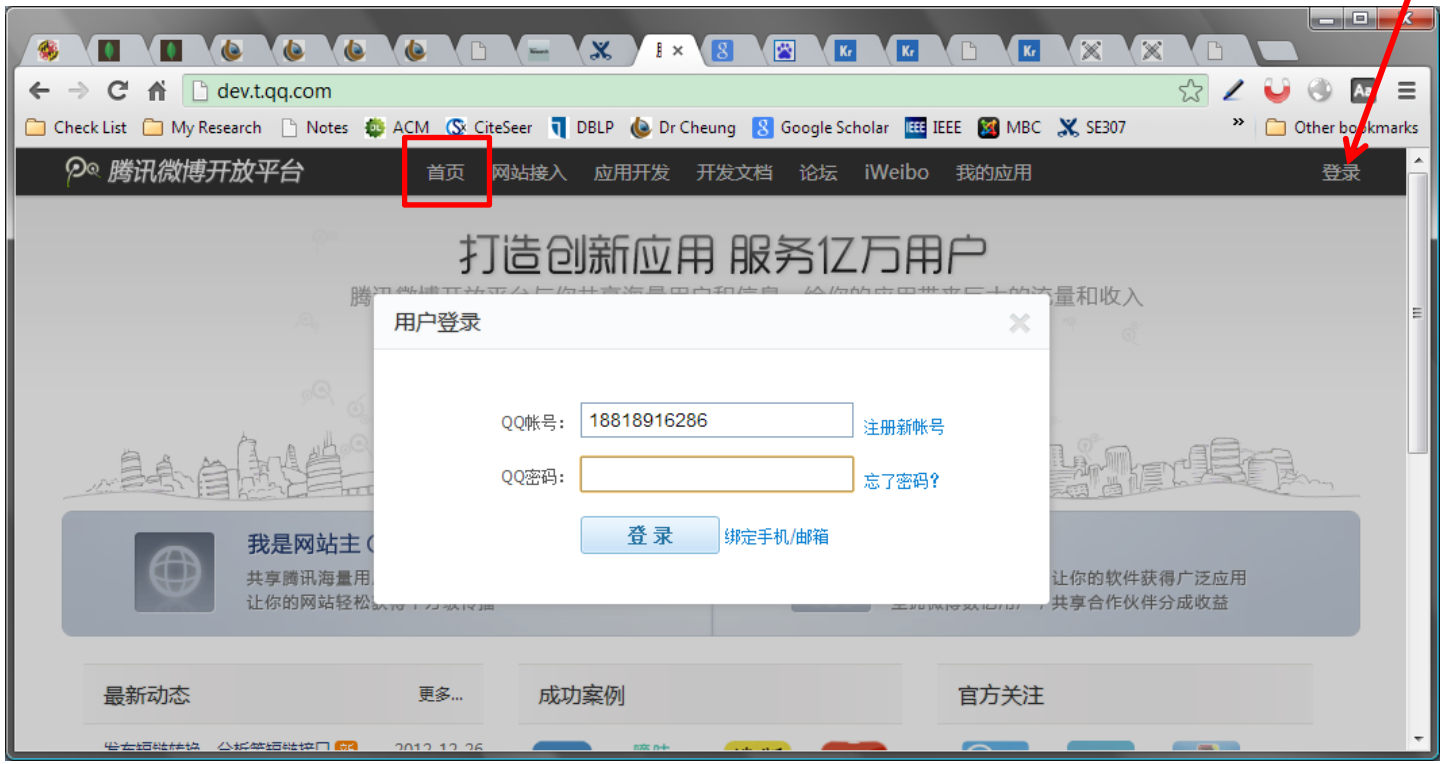


# Functionality failure

- Open a file you have no right to write.
- The temporary file under %temp% override the current file content

# Functionality failure

Click it?
Login dialog open



# Functionality failure

Click it?
No response!

# Usability

- Minimize the effort needed for use of the software

## Usability disasters

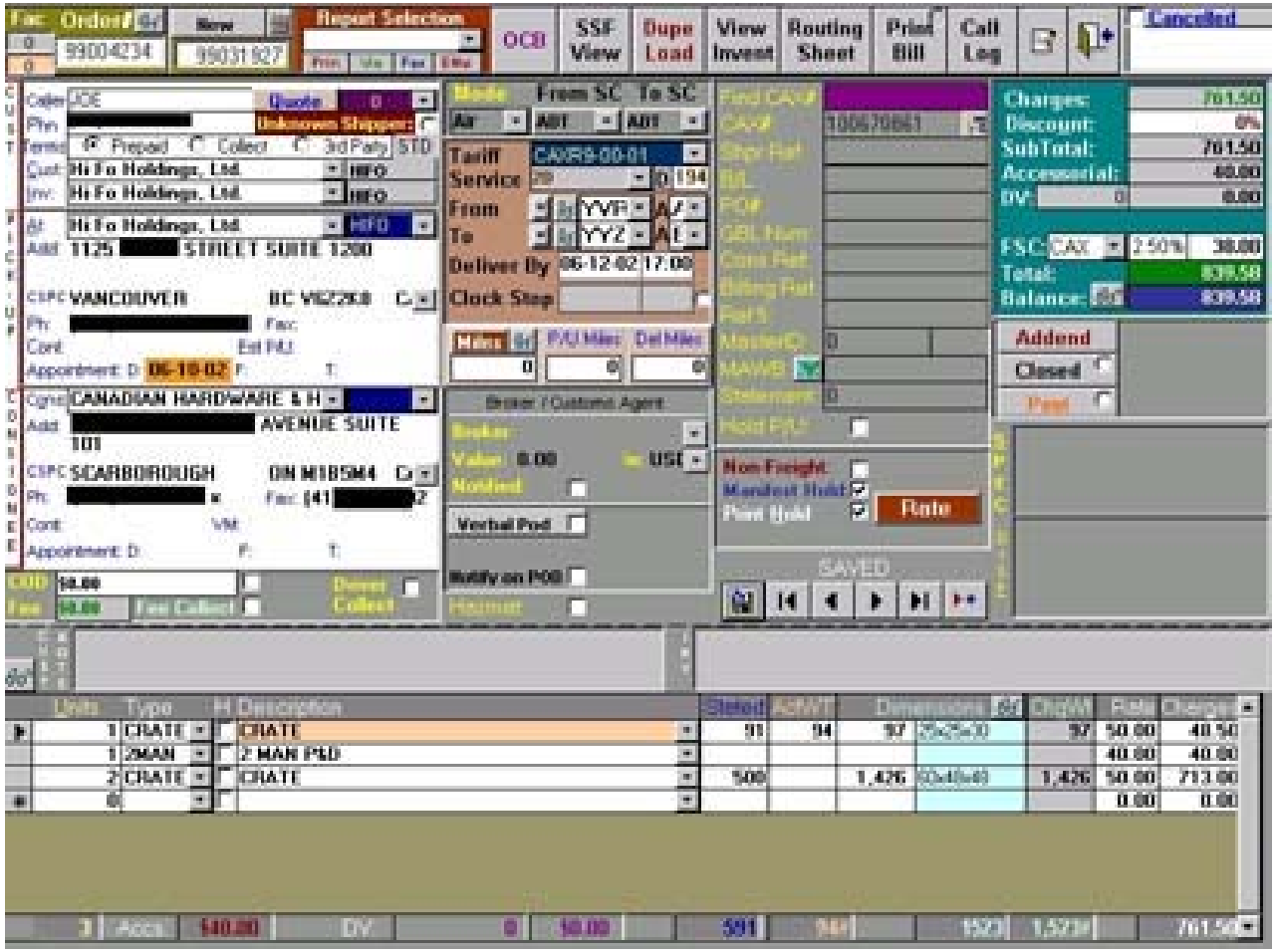



# Usability

- **Understandability**: The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
- **Learnability**: The capability of the software product to enable the user to learn its application
- **Operability**: The capability of the software product to enable the user to operate and control it.
- **Likeability**: The capability of the software product to be liked by the user.

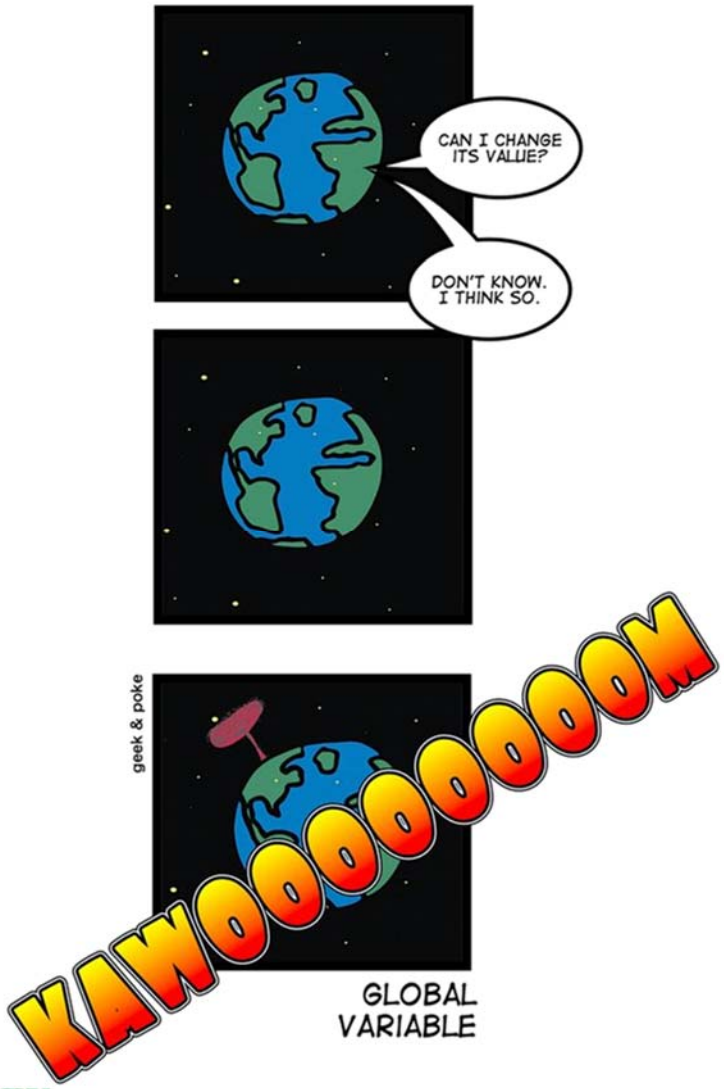
# Usability

- Don't!



# Reliability

- The probability of failure free operation of a computer program in a specified environment for a specified time.
  - MTTF: mean time to failure



# Reliability

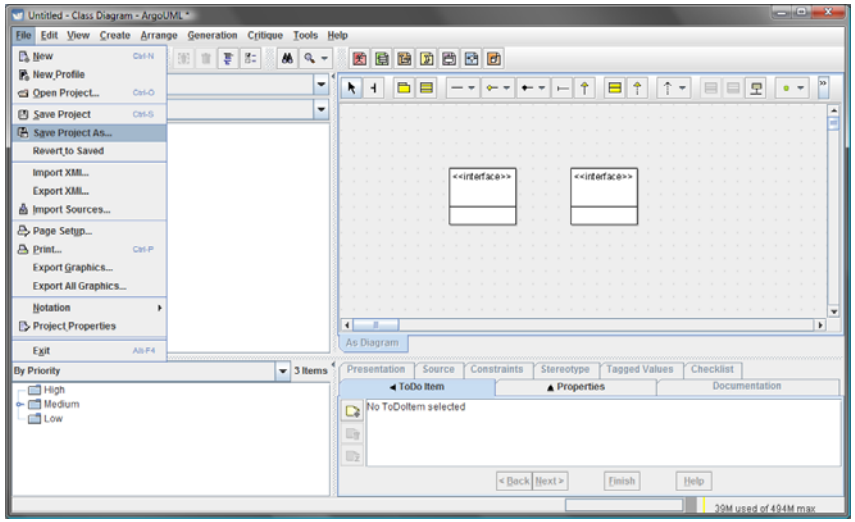
- **Availability:** The capability of the software to be in a state to perform a required function at a given point in time, under stated conditions of use.
- **Fault tolerance**: The capability of the software to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
- **Recoverability:** The capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure.

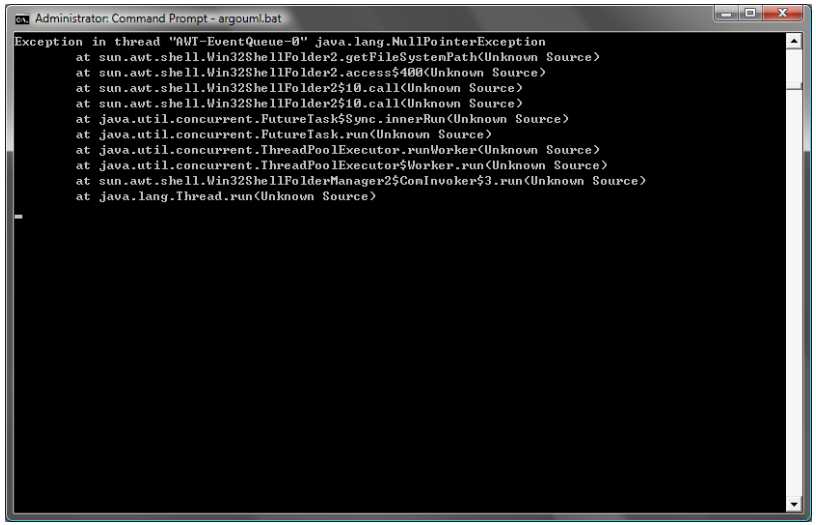
# Reliability

- **Robustness**: the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

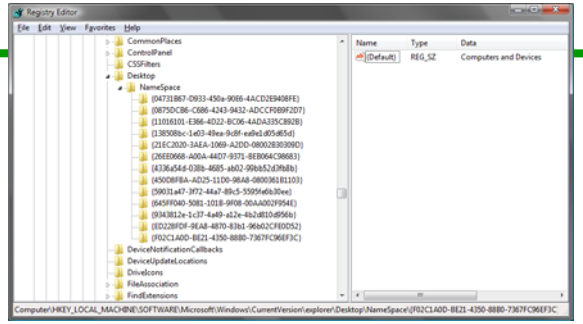
# Reliability

- A real world robustness bug



# Reliability



- A real world robustness bug

Root cause: a piece of code in JDK 1.7: sun.awt.shell.Win32ShellFolder2.java

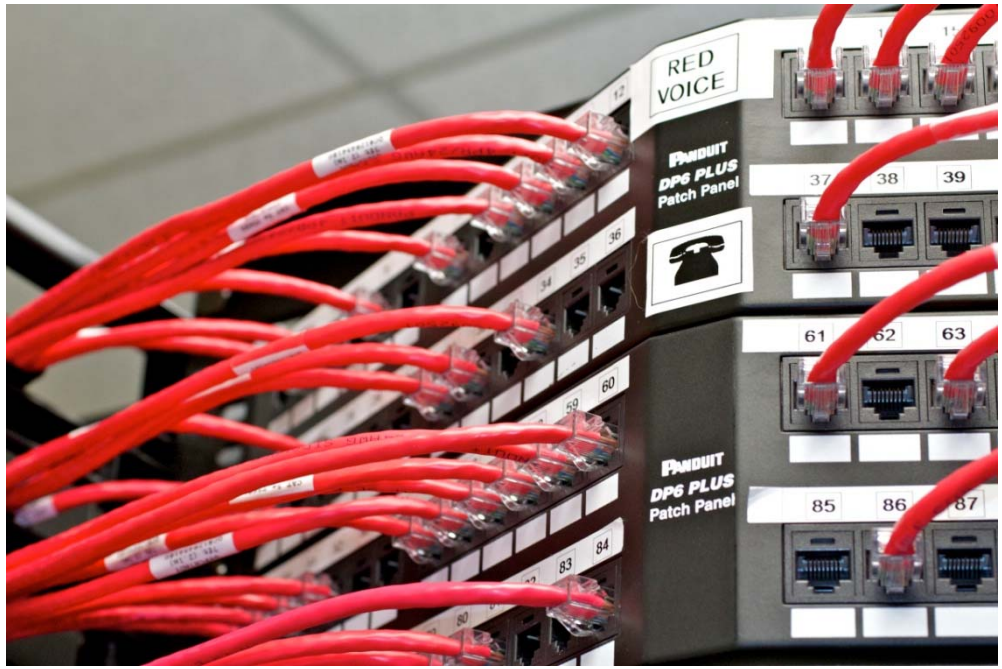
```
// Return the path to the underlying file system object
private static String getFileSystemPath(final long parentIShellFolder, final long relativePIDL) {
    return invoke(new Callable<String>() {
        public String call() {
            int linkedFolder = ATTRIB_LINK | ATTRIB_FOLDER;
            if (parentIShellFolder == Win32ShellFolderManager2.getNetwork().getIShellFolder() &&
                getAttributes0(parentIShellFolder, relativePIDL, linkedFolder) == linkedFolder) {

                String s =
                    getFileSystemPath(Win32ShellFolderManager2.getDesktop().getIShellFolder(),
                                      getLinkLocation(parentIShellFolder, relativePIDL, false));
                if (s != null && s.startsWith("\\\\")) {
                    return s;
                }
            }
            return getDisplayNameOf(parentIShellFolder, relativePIDL, SHGDN_FORPARSING);
        }
    });
}
```

# Maintainability

- The ease of maintaining the software

Unmaintainable

Maintainable



# Maintainability

The capability of the software to be modified.

Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

# Maintainability

- **Changeability**: The capability of the software product to enable a specified modification to be implemented**.**
- **Stability:** The capability of the software to minimise unexpected effects from modifications of the software
- **Testability**: The capability of the software product to enable modified software to be validated.

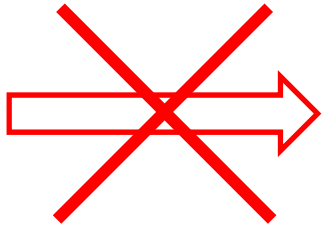
# Portability

The capability of software to be transferred from one environment to another.

The environment may include organisational, hardware or software environment.

# Interoperability

The capability of a system to work with other systems

# Software Quality Factors

Maintainability

Portability

Interoperability

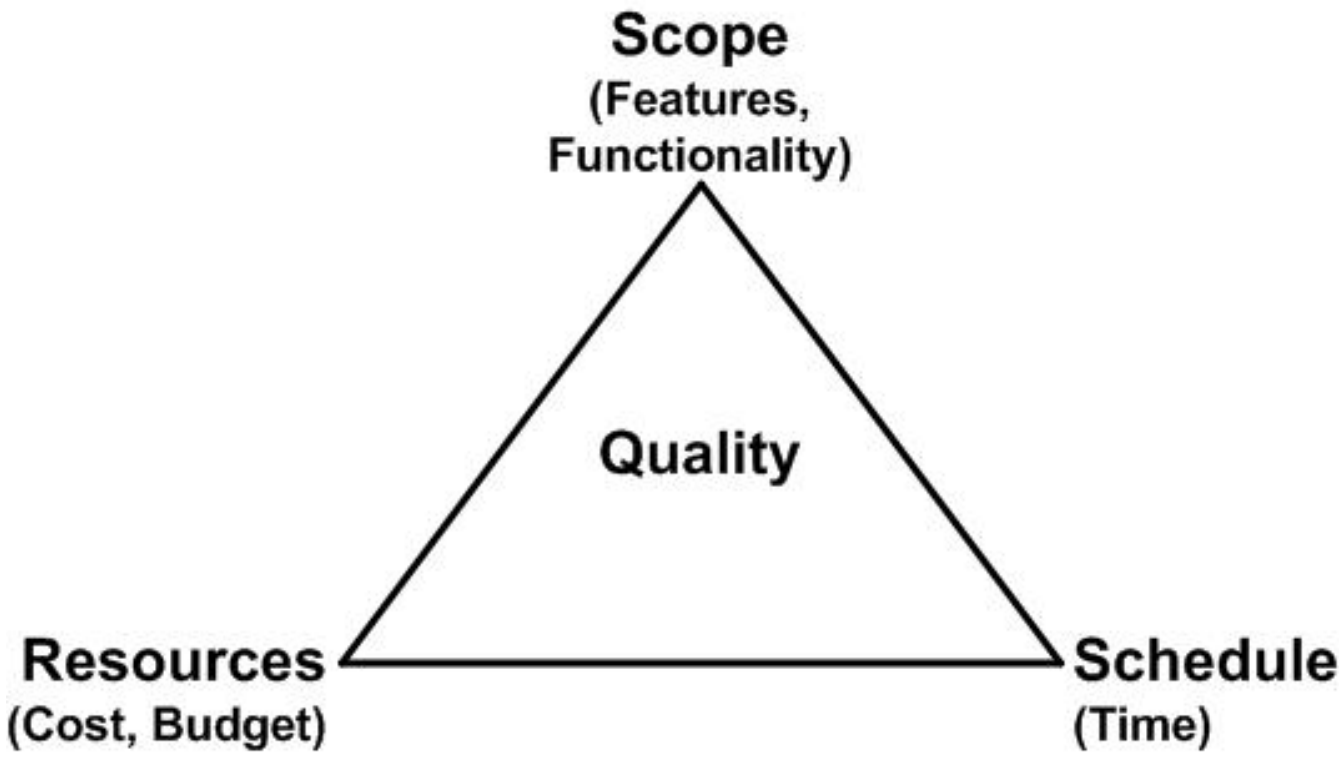

Functionality, Reliability

Efficiency, Usability

# Software quality has became an important part of software engineering

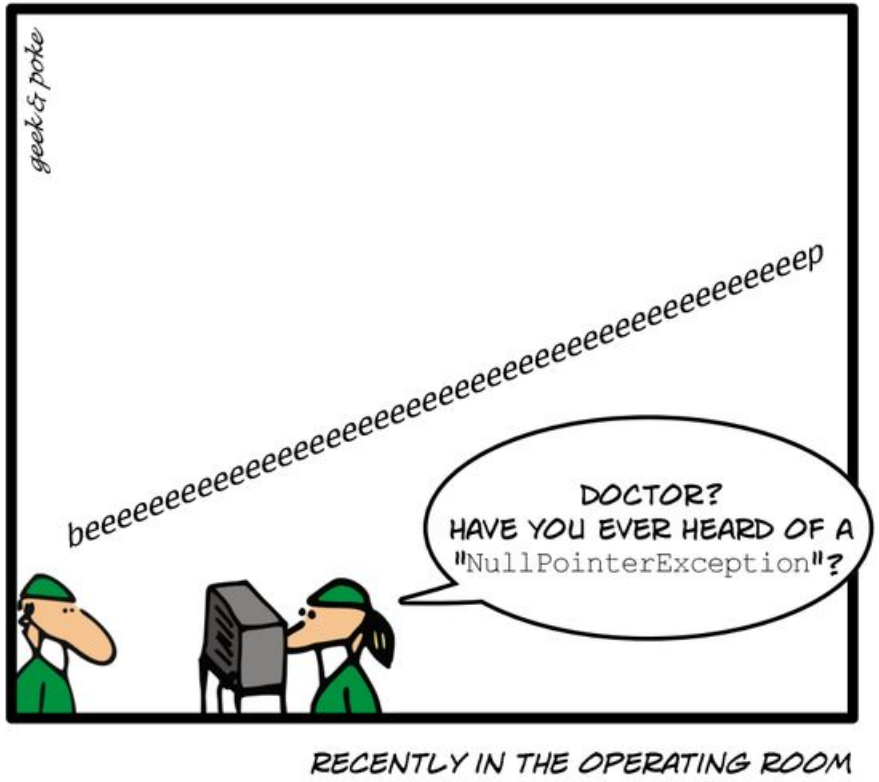
- A brief history of software engineering:
  - The **functional** era (1960 and earlier)
    - How to make the program?
    - "*23% of all projects are cancelled before completion.*"
  - The **schedule** era (1970s)
    - How to make the program before deadline?
    - "*Only 28% of projects are completed on time*"
  - The **cost** era (1980s)
    - How to make the program before the deadline at the budgeted cost?
    - "*49% of projects cost 189% of their original estimates.*"
  - The **quality** era (1990s and today)
    - How to make the program before the deadline at the budgeted cost and with good quality?

# What we know now: the "iron triangle" rule of software development



# The bane of quality: software bugs

- Have you heard of software bug?




# Concepts: Failure, Fault and Error

- **Failure**
  Observable incorrect behavior or state of a given system.

- **Fault (also known as “bug” and “defect”)**
  A defect in a system whose presence causes the failure.

- **Error**
  The human mistake that produce a fault.

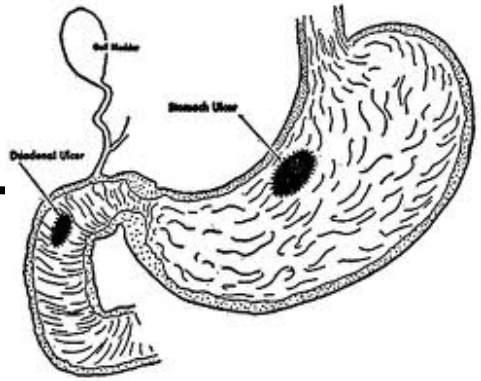

can lead to

can lead to


Error Fault Failure

# Concepts: Failure, Fault and Error

*Error* *Fault* *Failure*



# Example …

| LOC | Code |
|---|---|
| 1 | program double (); |
| 2 | var x,y: integer; |
| 3 | begin |
| 4 | read(x); |
| 5 | y := x * x;  // Shall be  y := x + x; |
| 6 | write(y) |
| 7 | end |

*Failure*:  x = 3 means y =9 →Failure!

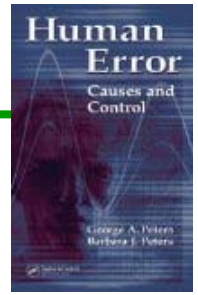
- This is a failure of the system since the correct output would be *6*

*Fault*: The fault that causes the failure is in line 5. The * operator is used instead of +.

*Error*: The error that conduces to this fault may be:

- a typing error (the developer has written * instead of +)
- a conceptual error (e.g., the developer doesn't know how *to double a number*)

# Where are bugs coming from?




- Who did it? Whose fault? Who shall be blamed?
  - **Client:** Inappropriate software reuse, communication error.
  - **Requirement engineer**: Omission, misunderstanding, or deliberate deviations of software requirements.
  - **Designer**: Wrong algorithms, wrong handling of boundary conditions, omission of system states, missing abnormal operation handling.
  - **Programmer**: Misunderstanding of design doc, mental lapse while coding, non-compliance with documentation and coding instructions.
  - **Quality assurance team**: Incomplete test plan, not document and report detected faults or failures, not promptly correct faults, incomplete correction of detected errors.
  - **Operational staff**: Procedure errors.
  - **Technical writer**: Omission of software functions, not up-to-date, error in descriptions.

# Fault classification: ODC*

Distribution among 12 open sources projects (e.g. Linux kernal) over 600+ reported fault

| ODC types | Nature | Examples | # faults | % of total |
|---|---|---|---|---|
| Assignm. | Missing | A variable was not assigned a value, a variable was not initialized, etc | 62 | 9.3 % |
| | Wrong | A wrong value (or expression result, etc) was assigned to a variable | 70 | 10.5 % |
| | Extraneous | A variable should not have been subject of an assignment | **11** | 1.6 % |
| Checking | Missing | An "if" construct is missing, part of a logical condition is missing, etc | 113 | 16.9 % |
| | Wrong | Wrong logical expression used in a condition in brach and loop onstruct (if, while, etc.) | 53 | 7.9 % |
| | Extraneous | An "if" construct is superfluous and should not be present | 1 | 0.1 % |
| Interface | Missing | A parameter in a function call was missing; incomplete expression was used as param. | 11 | 1.6 % |
| | Wrong | Wrong information was passed to a function call (value, expression result etc) | 38 | 5.7 % |
| | Extraneous | Surplus data is passed to a function (e.g. one parameter too many in function call) | 0 | 0 % |
| Algorithm | Missing | Some part of the algorithm is missing (e.g. function call, a iteration construct, etc) | 222 | 33.2 % |
| | Wrong | Algorithm is wrongly coded or ill-formed | 40 | 6 % |
| | Extraneous | The algorithm has surplus steps; A function was being called | 6 | 0.9 % |
| Function | Missing | New program modules were required | 21 | 3.1 % |
| | Wrong | The code structure has to be redefined to correct functionality | 20 | 3 % |
| | Extraneous | Portions of code were completely superfluous | 0 | 0 % |

*R. Chillarege, “Orthogonal Defect Classification,” Handbook of Software Reliability Eng., chapter 9, IEEE CS Press, McGraw-Hill, 1995.

# Fault #1

- Mars Climate Orbiter (1999-11-10)
  - Purpose: to relay signals from the Mars Polar Lander once it reached the surface of the planet
  - smashed into the planet instead of reaching a safe orbit
  - The fault (wrong assignment): fail to convert English measures to metric values


http://www.cnn.com/TECH/space/9911/10/orbiter.02/

# Fault #2

- European Space Agency Ariane 5 Flight 501 (June 4, 1996)
  - Self-destruction 40 seconds after takeoff
  - It took 10 years and $7 billion to build
  - The fault (missing check): a conversion from 64-bit floating point to 16 bit integer with a value larger than possible. The overflow caused a hardware trap.

# Fault #3

- The 2003 North America blackout
  - Triggered by a local outage that went undetected.
  - The fault (missing function call): a *race condition* in General Electric's monitoring software prevented an alarm.


Democrat and Chronicle

Huge power failures

Six pages of coverage

No delays for PGA

BLACKOUT

Affected cities across the Northeast

# Fault #4

- The MIM-104 Patriot (February 25, 1991)
  - Fail to intercept an incoming missile, deaths of 28 Americans in Dharan, Saudi Arabia
  - The radar traces the trajectory model in real time. ("in x micro-second the missile should be in position y")
  - The fault (wrong algorithm): due to rounding of the clock values, it accumulates inaccuracies. After several hours this inaccuracy is critical.

# Fault #5

- 東涌纜車電腦程式故障 2006-06-18

  "The operator of the Ngong Ping 360 cable car says it's not confident that the ride can be opened to the public as scheduled this coming Saturday. A trial run yesterday left 500 passengers stranded in mid-air for two hours. Chairman of Skyrail-ITM, Ken Chapman, said ***the problem was caused by an error in the programming system***, and had nothing to do with the safety of the ride. Mr Chapman apologized for the inconvenience caused."

http://www.rthk.org.hk/rthk/news/englishnews/subframe.htm?20060618&56&318652

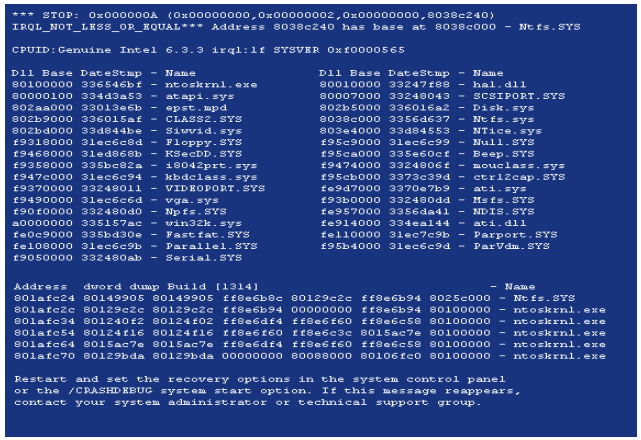
# Fault #6

- 温州动车事故 2010-7-23
  - “从软件及系统设计看，温州南站使用的**LKD2-T1**型列控中心保险管**F2**熔断后，采集驱动单元检测到采集电路出现故障，向列控中心主机发送故障信息，但未按“故障导向安全”原则处理采集到的信息，导致传送给主机的状态信息一直保持为故障前采集到的信息；列控中心主机收到故障信息后，仅把故障信息转发至监测维护终端，也未采取任何防护措施，继续接收采集驱动单元送来的故障前轨道占用信息，并依据故障前最后时刻的采集状态信息控制信号显示及轨道电路。”

http://news.sina.com.cn/c/2011-12-29/024223712039.shtml

# Industry Standard on Faults

- Carefully made programs:
  - 30 to 85 defects per 1K of code during development
  - ½ to 3 defects per 1K of code latent in the delivered product
  - 1M LOC will have 3000 faults




- Example: Windows 2000
  - Over 29M LOC
  - Theoretically: 29 x 3000 = 87,000 faults
  - Actually: 63,000 faults at the first release (according to an internal memo obtained by CNN*)

Beijing 2008 Olympics

*: 17 Feb 2000, Will bugs scare off users of new Windows 2000 - CNN
http://articles.cnn.com/2000-02-17/tech/windows.2000_1_microsoft-memo-windows-nt-software-washingtonbased-microsoft?_s=PM:TECH

# Microsoft EULA

- **11**. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, **IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER** (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) **ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT**, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

# GPL

- **11.** BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. **THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU**. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
- **12. IN NO EVENT** UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING **WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM** AS PERMITTED ABOVE, **BE LIABLE TO YOU FOR DAMAGES**, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES **ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM** (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

# Discussion

- Share your hairiest bug war stories.

CACM, 1997, Vol. 40, No. 4


My Hairiest Bug War Stories

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

# Thank you!