Part V 除錯 (Debugging)

Shin-Jie Lee (李信杰)
Associate Professor
Computer and Network Center
Department of CSIE
National Cheng Kung University

What is BUG?

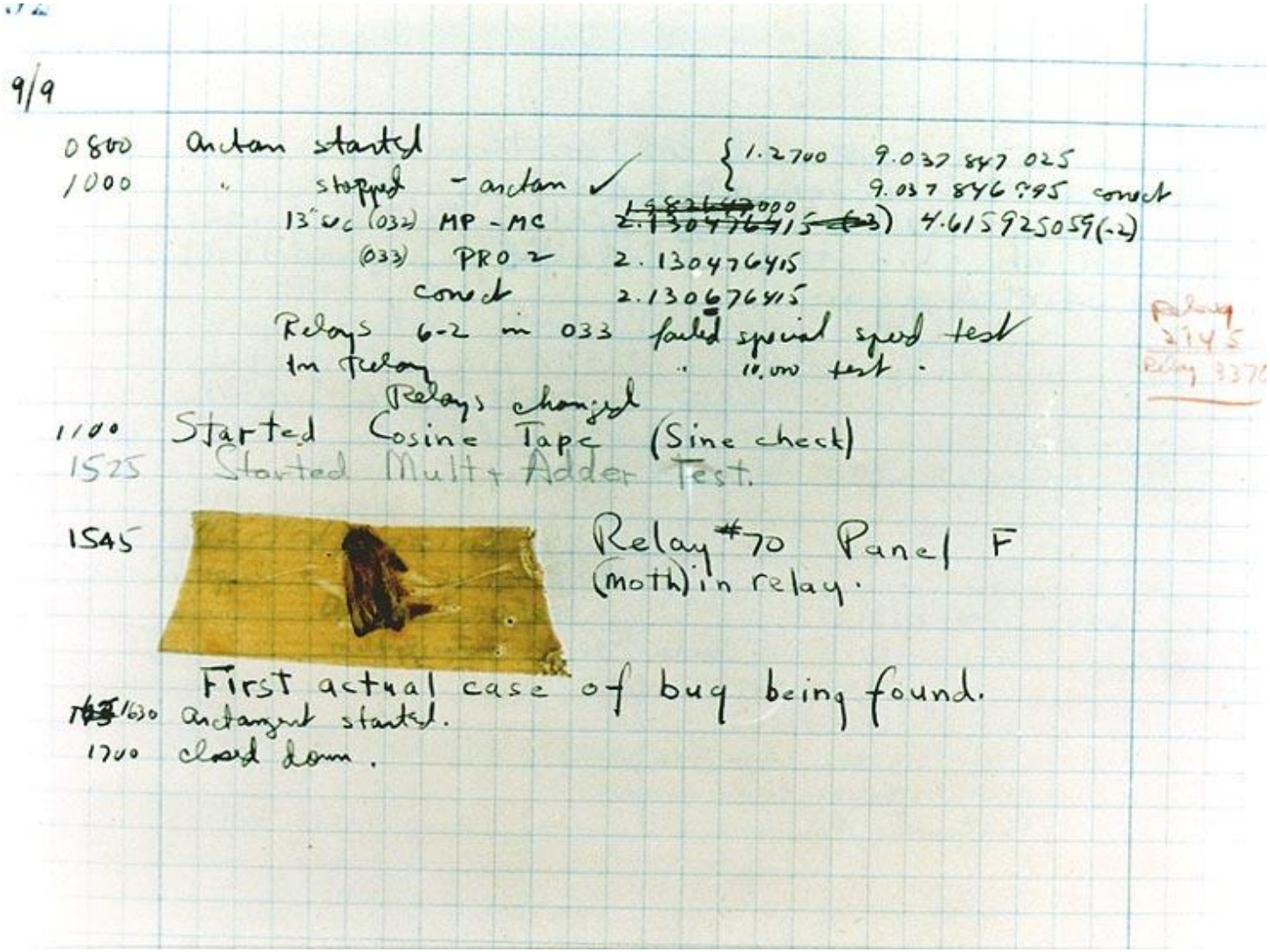
- Things the software does that it is not supposed to do, [or] something the software doesn't do that it is supposed to. [Telles and Hsieh]

- A **software bug** is an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways. [From Wikipedia]

- 1. Synonym of *defect*. 2. Synonym of *failure*. 3. Synonym of *problem*. 4. Synonym of *infection*. [Andreas Zeller]

[Telles and Hsieh] Telles, Matt and Yuan Hsieh. *The Science of Debugging*. Scottsdale: Coriolis, 2001.
[Andreas Zeller] Andreas Zeller. Why Programs Fail, Second Edition: A Guide to Systematic Debugging, 2009

The First "Computer Bug" Moth found trapped between points at Relay # 70, Panel F, of the Mark II Aiken Relay Calculator while it was being tested at Harvard University, 9 September 1947.



Bug-Free Software?

沒有效率的除錯方式1

- **Find the defect by guessing (用猜的)**
  - 再程式碼中隨機插入print
  - 如果無法找到錯誤，則試著改東改西直到可以work
  - 沒有備份
  - 程式越改越亂

沒有效率的除錯方式2

- **Don't waste time trying to understand the problem (不花點時間了解問題)**
  - 也許問題不難，甚至不需要完全了解它就解了，但就是不願花點時間了解

沒有效率的除錯方式3

- **Fix the error with the most obvious fix (哪裡有洞修哪裡、東補西補沒有治根)**
  - 只修正眼前表面的錯誤，而不花時間找出根本的問題

```
x = compute(y)
If(y==17)
    x=25.15   -- compute() doesn't work for y=17, so fix
it
```

沒有效率的除錯方式4

- **Debugging by Superstition (覺得自己寫的程式都沒有問題)**
  - 如果你遇到一個你寫的程式問題，那是你的錯，不是電腦的錯、也不是Compiler的錯
  - 即使問題表面上應該不是你寫的程式造成的問題，但強烈建議先假設是你的問題
  - **當你先入為主認為不是你的問題，那麼這個Bug將非常難被找到**

Tips for Finding Defects$_1$

- **Use all the data available to make your hypothesis (提出假設、不是亂猜)**
  - When creating a hypothesis about the source of a defect, account for as much of the data as you can in your hypothesis
- **Refine the test cases that produce the error (重現問題)**
  - You might be able to vary one parameter more than you had assumed, and focusing on one of the parameters might provide the crucial breakthrough

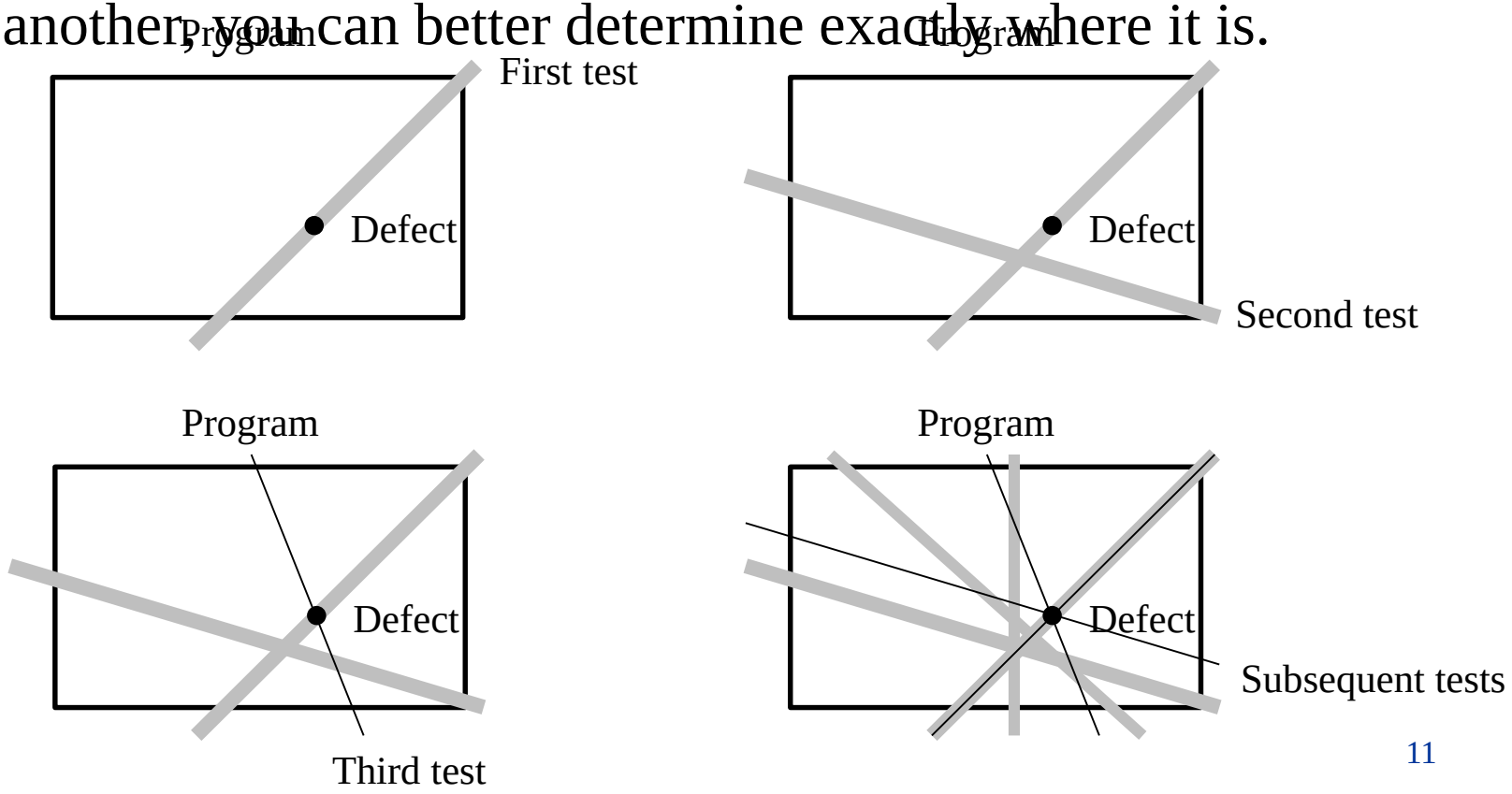
Tips for Finding Defects$_2$

- **Exercise the code in your unit test suite (重跑單元測試)**
  - Defects tend to be easier to find in small fragments of code than in large integrated programs. Use your unit tests to test the code in isolation.

- **Use available tools (使用除錯工具)**
  - With one tough-to-find error, for example, one part of the program was overwriting another part's memory.
    - This error was difficult to diagnose using conventional debugging practices.
  - To use tool, for example, PyCharm.

Tips for Finding Defects$_3$

- **Reproduce the error several different ways (旁敲側擊)**
  - If you can get a fix on it from one point and a fix on it from another, you can better determine exactly where it is.



Tips for Finding Defects$_4$

- **Generate more data to generate more hypotheses (使用更多的測試資料進而提出更多的假設)**
  - Choose test cases that are different from the test cases you already know to be erroneous or correct.
  - Run them to generate more data, and use the new data to add to your list of possible hypotheses.

- **Use the results of negative tests (一步步排除假設)**
  - Suppose that a test case disproves your hypothesis, so you still don't know the source of error.
  - However, you do know that the defect is not in the area you thought it was. That narrows your search field and the set of remaining possible hypotheses.

Tips for Finding Defects$_5$

- **Brainstorm for possible hypotheses (腦力激盪)**
  - Rather than limiting yourself to the first hypothesis you think of, try to come up with several

- **Keep a notepad by your desk, and make a list of things to try (將可能的想法筆記下來)**
  - One reason programmers get stuck during debugging sessions is that they go too far down dead-end paths.
  - Make a list of things to try, and if one approach isn't working, move on to the next approach

Tips for Finding Defects$_6$

- **Narrow the suspicious region of the code (二分法)**
  - Rather than removing regions haphazardly, divide and conquer
  - Use a binary search algorithm to focus your search

Tips for Finding Defects 7

- **Check code that's changed recently (檢查最近改的 Code)**
  - If you can't find a defect, run an old version of the program to see whether the error occurs
  - Check the version control log to see what code has changed recently

- **Expand the suspicious region of the code (擴大可疑程式碼聚焦)**
  - If you don't find the defect in a focused small section of code, consider the possibility that the defect isn't in the section

Tips for Finding Defects$_8$

- **Integrate incrementally (漸進式整合測試)**
  - If you add a piece to a system and encounter a new error, remove the piece and test it separately

Tips for Finding Defects$_{10}$

- **Talk to someone else about the problem (找人聊聊你的問題、整理思緒)**
  - You often discover your own defect in the act of explaining it to another person
- **Take a break from the problem (停止思考、休息一下吧!)**
  - Sometimes you concentrate so hard you can't think
  - The auxiliary benefit of giving up temporarily is that it reduces the anxiety associated with debugging