ECE453/SE465/CS447/ECE653/CS647:

Fault, Error, Failure

Lin Tan

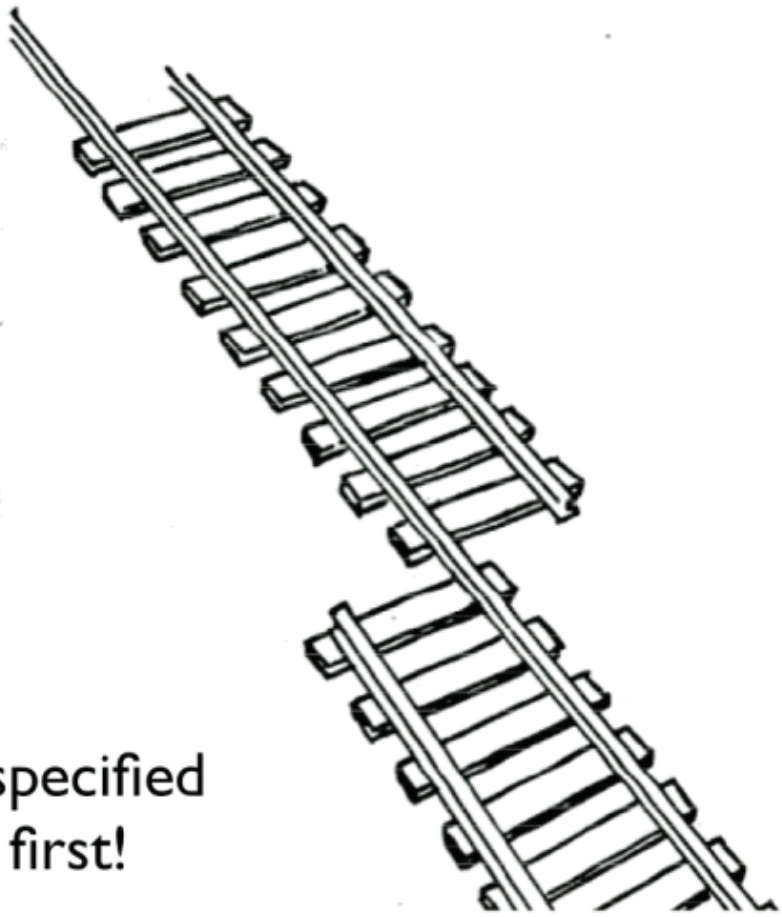
January 3, 2016

Terminology, IEEE 610.12-1990

- Fault -- often referred to as Bug *[Avizienis'00]*
  - A static defect in software (incorrect lines of code)
- Error
  - An incorrect internal state (*unobserved*)
- Failure
  - External, incorrect behaviour with respect to the expected behaviour (*observed*)

- Not used consistently in literature!

What is this?



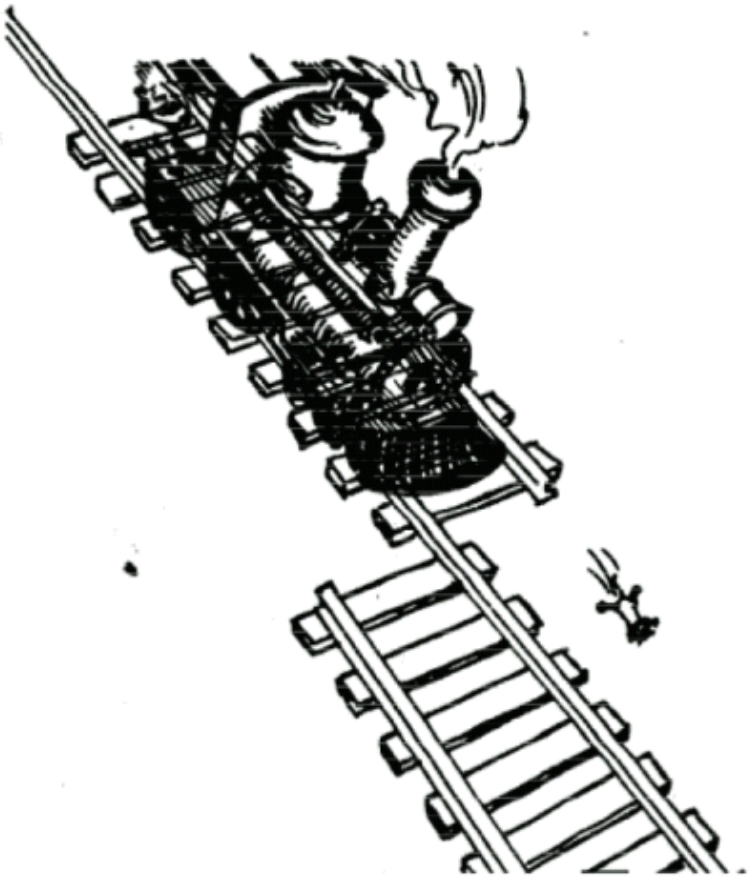
A failure?

An error?

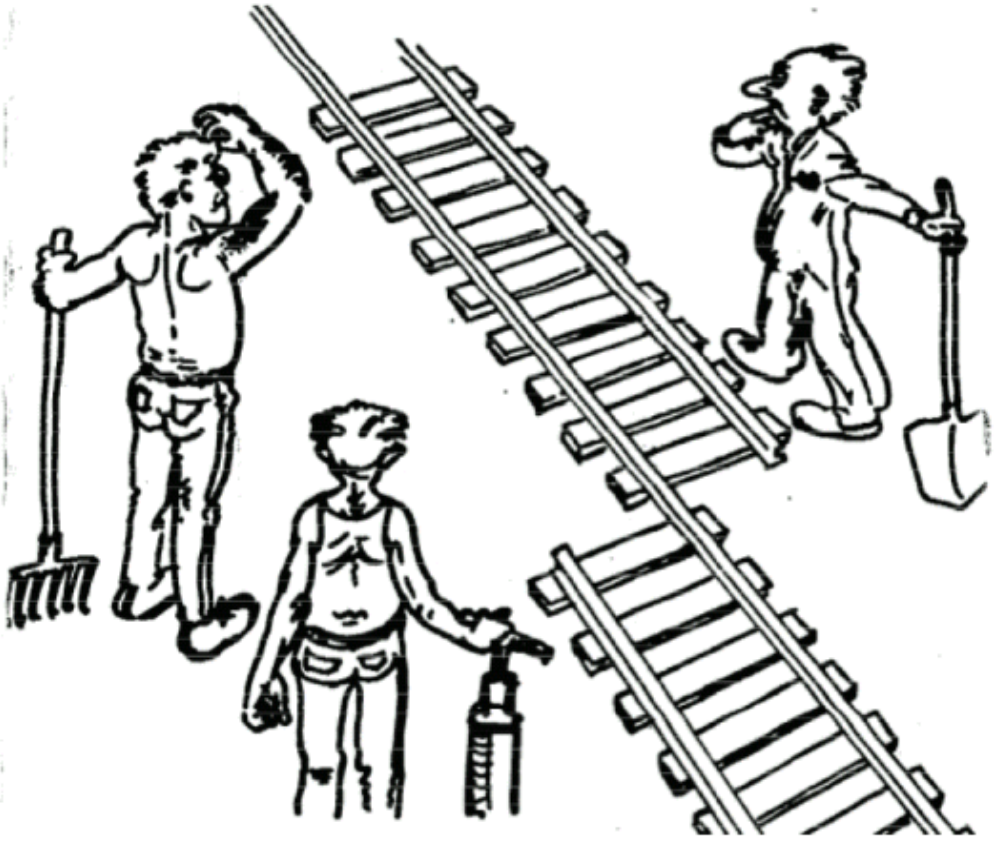
A fault?

We need to describe specified and desired behaviour first!

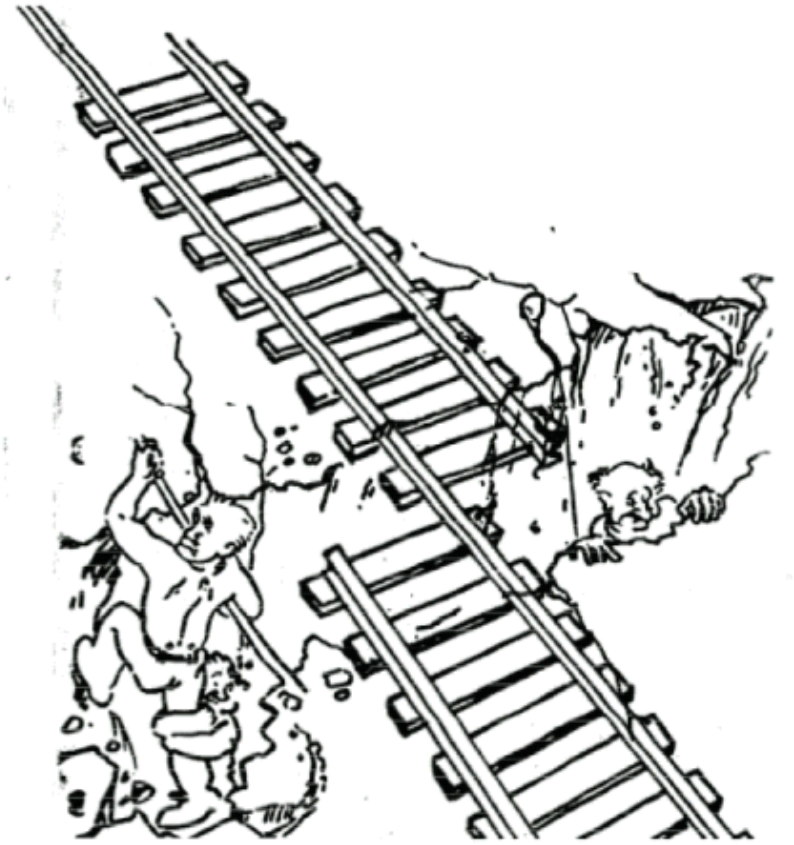
Erroneous State ("Error")



Design Fault



Mechanical Fault



Example: Fault, Error, Failure

```
public static int numZero (int[] x) {
//Effects: if x==null throw NullPointerException
//          else return the number of occurrences of 0 in x
   int count = 0;
   for (int i =1; i <x.length; i++) {
      if (x[i]==0) {
         count++;
      }
   }
   return count;
}
```

Wrong State:
x = [2,7,0]
i = 1
count =0
PC=first iteration of if

Expected State:
x = [2,7,0]
i =0
count =0
PC=first iteration of if

Fix: for (int i =0, i<x.length; i++)

x = [2,7,0], fault executed, error, no failure

x = [0,7,2], fault executed, error, failure

State of the program: x, i, count, PC

Exercise

```
public int findLast (int[] x, int y)
{
//Effects: If x==null throw NullPointerException
//   else return the index of the last element
//   in x that equals y.
//   If no such element exists, return -1
   for (int i=x.length-1; i > 0; i--)
   {
      if (x[i] == y)
      {
         return i;
      }
   }
   return -1;
}
   // test:  x=[2, 3, 5]; y = 2
   //        Expected = 0
```

Exercise - cont.

- Read this faulty program, which includes a test case that results in failure. Answer the following questions.
  - (a) Identify the fault, and fix the fault.
  - (b) If possible, identify a test case that does not execute the fault.
  - (c) If possible, identify a test case that executes the fault, but does not result in an error state.
  - (d) If possible identify a test case that results in an error, but not a failure. Hint: Don't forget about the program counter.
  - (e) For the given test case, identify the first error state. Be sure to describe the complete state.

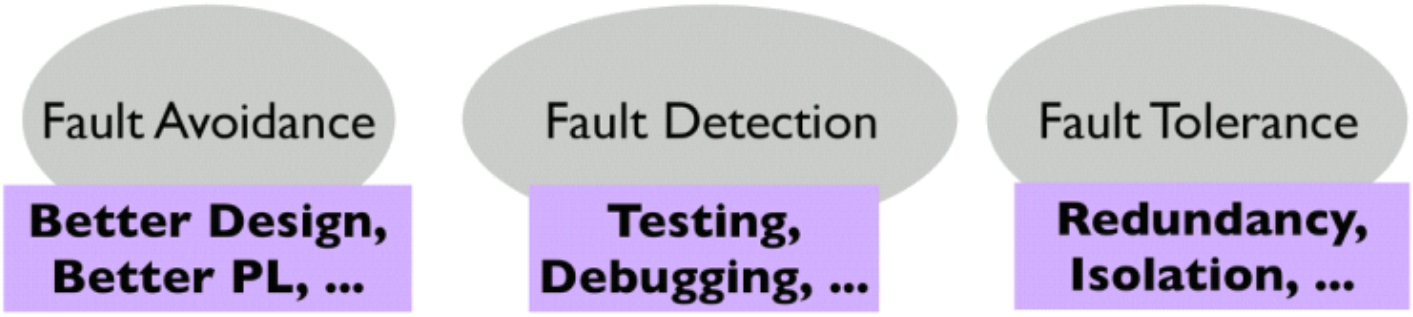
e) x = [2, 3, 5]
y = 2
i = 0 (undefined)
PC = return -1;

RIP Model

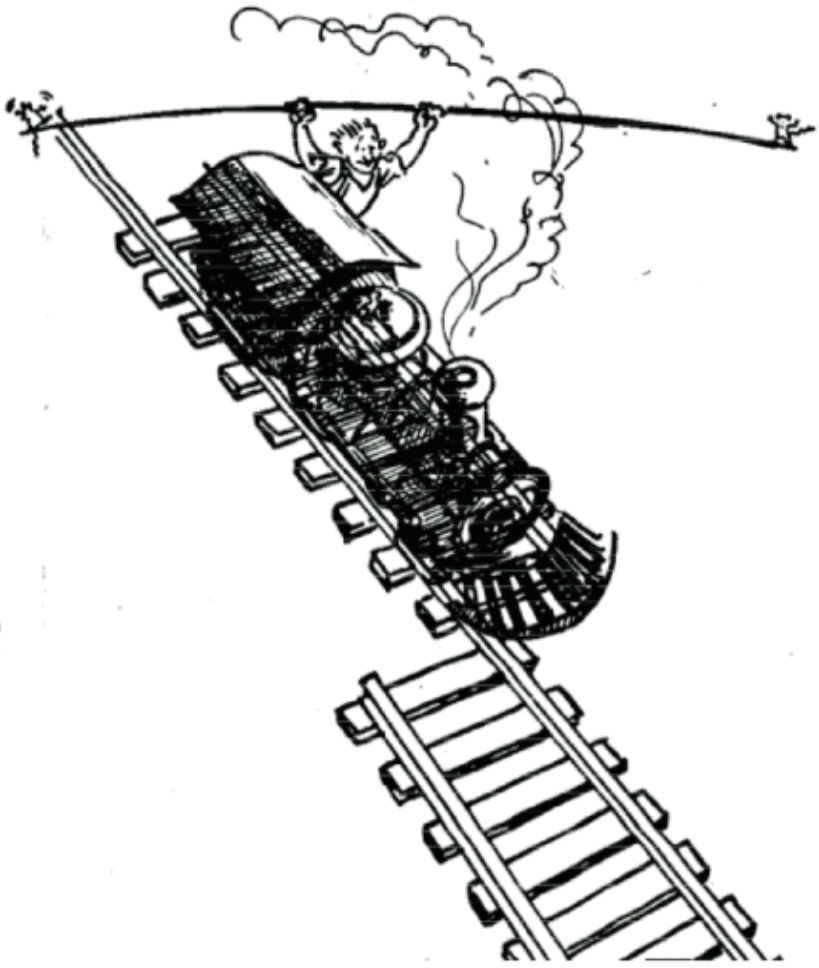
- Three conditions must be present for a failure to be observed:
  - Reachability: the location or locations in the program that contain the fault must be reached.
  - Infection: After executing the location, the state of the program must be incorrect.
  - Propagation: The infected state must propagate to cause some output of the program to be incorrect.

How do we deal with Faults, Errors, and Failures?

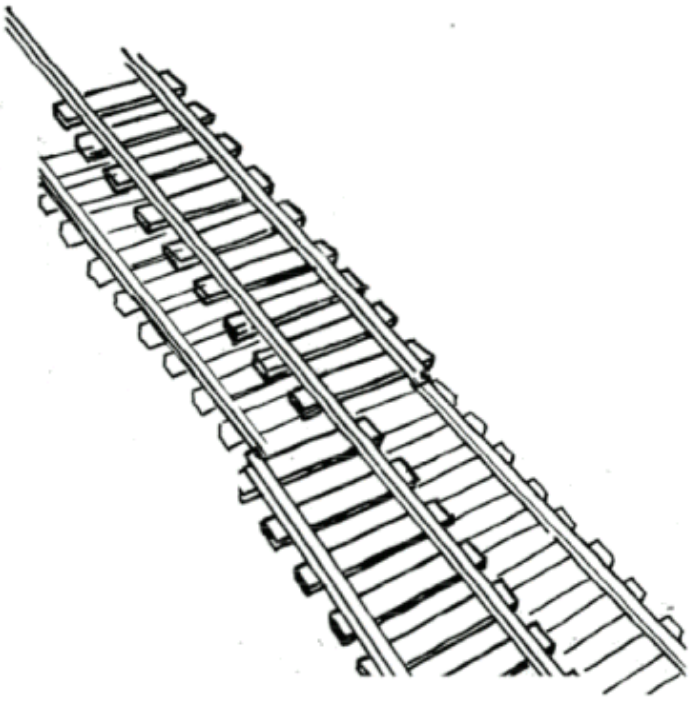
Addressing Faults at Different Stages



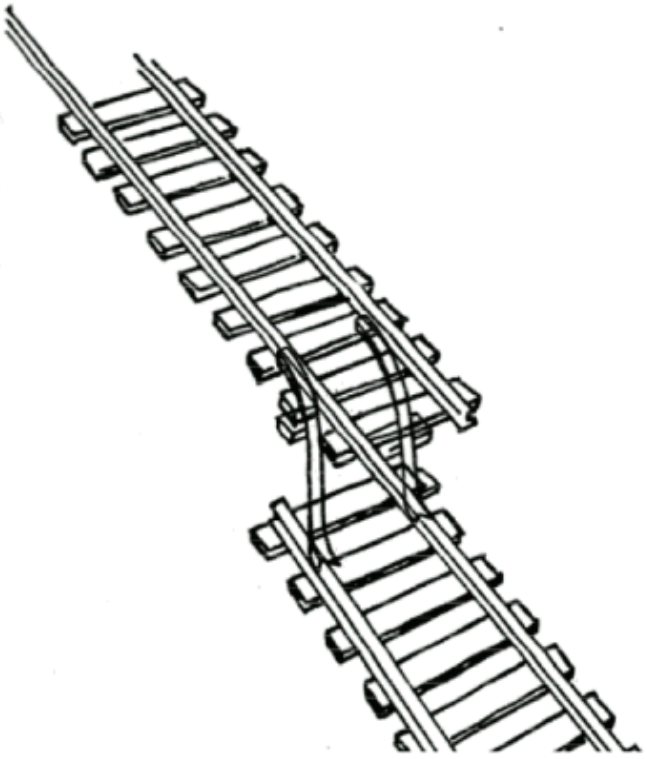
Declaring the Bug as a Feature



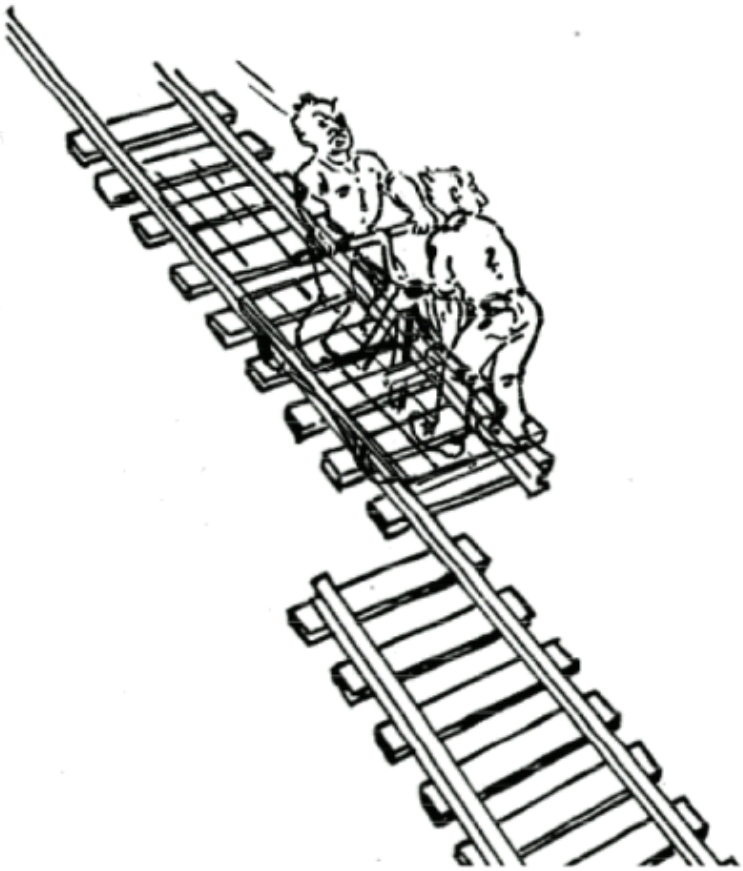
Modular Redundancy: Fault Tolerance



Patching: Fixing the Fault



Testing: Fault Detection



Testing vs. Debugging

- Testing: Evaluating software by observing its execution
- Debugging: The process of finding a fault given a failure

- Testing is hard:
  - Often, only specific inputs will trigger the fault into creating a failure.
- Debugging is hard:
  - Given a failure, it is often difficult to know the fault.

Testing is hard

```
if ( x - 100 <= 0 )
  if ( y - 100 <= 0 )
    if ( x + y - 200 == 0 )
      crash();
```

- Only input x=100 & y =100 triggers the crash.
- Probability to trigger the crash: 1 over 2^{64}
  - Assuming x and y are 32-bit integers