

## History of Software Testing



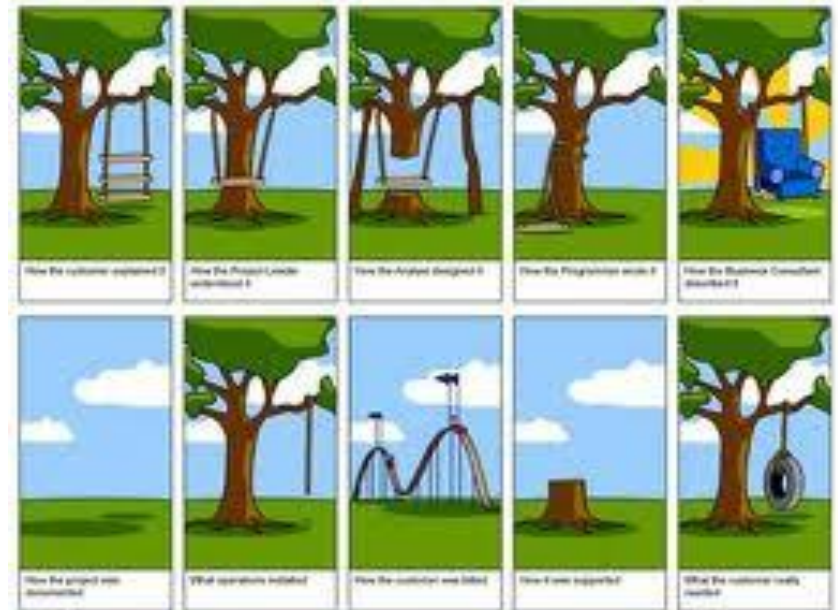
# Software Testing: Issues and Solutions

Asst. Prof. Dr. Jirapun Daengdej  
Faculty of Science and Technology  
Assumption University

# Topics of Interest & What Have We Done So Far?

# Topics


- Gathering requirements
- Analysis & Design
- Coding
- **Testing**
- **Installation & Maintenance**
- **Retirement**



# Welcome to Software Testing!!

# General Problems in Testing

**Complete testing?**  
What do we *mean* by "complete testing"?



- Complete "coverage": Tested every line / branch / basis path?
- Testers not finding new bugs?
- Test plan complete?

Complete testing **must** mean that, at the end of testing, you know there are no remaining unknown bugs.

- After all, if there are more bugs, you can find them if you do more testing. So testing couldn't yet be "complete."

Black Box Software Testing Copyright © 2003-5 *Om Khaner & James Bach*

***Exhaustive Testing is Impossible!!***

<http://www.qagenius.com/exhaustive-testing-is-impossible/>

**Concept of Complete Testing | Exhaustive testing is impossible!!**

<http://www.softwaretestingtimes.com/2010/04/concept-of-complete-testing-exhaustive.html>

# Yes, we have many tools available

## Support! different situations by specific tools



# Life is Not That Simple!!

What suits one customer  
might not suit the next



Regression:  
"when you fix one bug, you  
introduce several newer bugs."



# **Software Testing**

*What If I have a critical software??*

# Fact..

As the **120-ton space shuttle** sits surrounded by almost 4 million pounds of rocket fuel...

...the last three versions of the program -- each **420,000 lines long-had just one error each**. The last 11 versions of this software had a total of 17 errors...

<http://www.fastcompany.com/magazine/06/writestuff.html>

# Fact..

## Maybe NASA could use some buggy software

by JOHN on OCTOBER 8, 2009

In [Coders at Work](#), Peter Norvig quotes NASA administrator Don Goldin saying

We've got to do the better, faster, cheaper. These space missions cost too much. It'd be better to run more missions and some of them would fail but overall we'd still get more done for the same amount of money.

NASA has extremely rigorous processes for writing software. They supposedly develop bug-free code; I doubt that's true, though I'm sure they do have exceptionally low bug rates. But this quality comes at a high price. Rumor has it that space shuttle software costs \$1,500 per line to develop. When asked about the price tag, Norvig said "I don't know if it's optimal. **I think they might be better off with buggy software.**" At some point it's certainly not optimal. If it doubles the price of a project to increase your probability of a successful mission from 98% to 99%, it's not worth it; you're better off running two missions with a 98% chance of success each.

Few people understand that software quality is all about probabilities of errors. Most people think the question is whether you'd rather have bug-free software or buggy software. I'd rather have bug-free software, thank you. But bug-free is not an option. Nothing humans do is perfect. All we can do is lower the probabilities of bugs. But **as the probability of bugs goes to zero, the development costs go to infinity.** (Actually it's not all about probabilities of errors. It's also about the consequences of errors. Sending back a photo with a few incorrect pixels is not the same as crashing a probe.)

# Our Focus

# Four Types of Testing

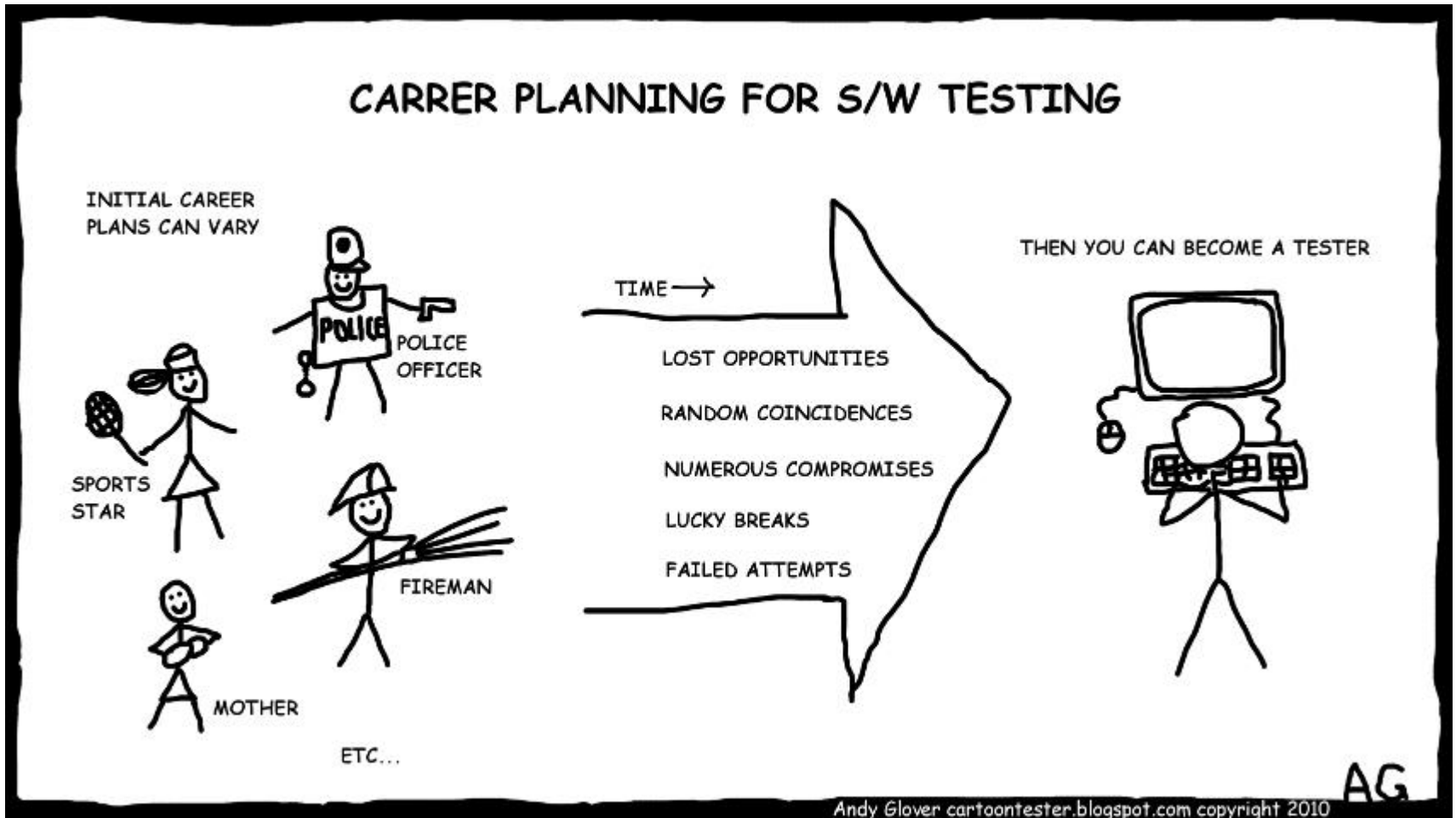
- According to the book “*Foundations of Software Testing*” by Aditya P. Mathur 2009
- There are 4 types of testing
  - Source of test generation.
  - Lifecycle phase in which testing takes place
  - Goal of a specific testing activity
  - Characteristics of the artifact under test

# Our Focus

- Source of test generation
  - Informal requirement/Code/Formal model
- Lifecycle phase in which testing takes place
  - Unit/Integration/System/Regression/...
- Goal of a specific testing activity
  - Special features/Security/Errors in GUI/...
- Characteristics of the artifact under test
  - Component/Client-Server/Database/Real-time SW/...

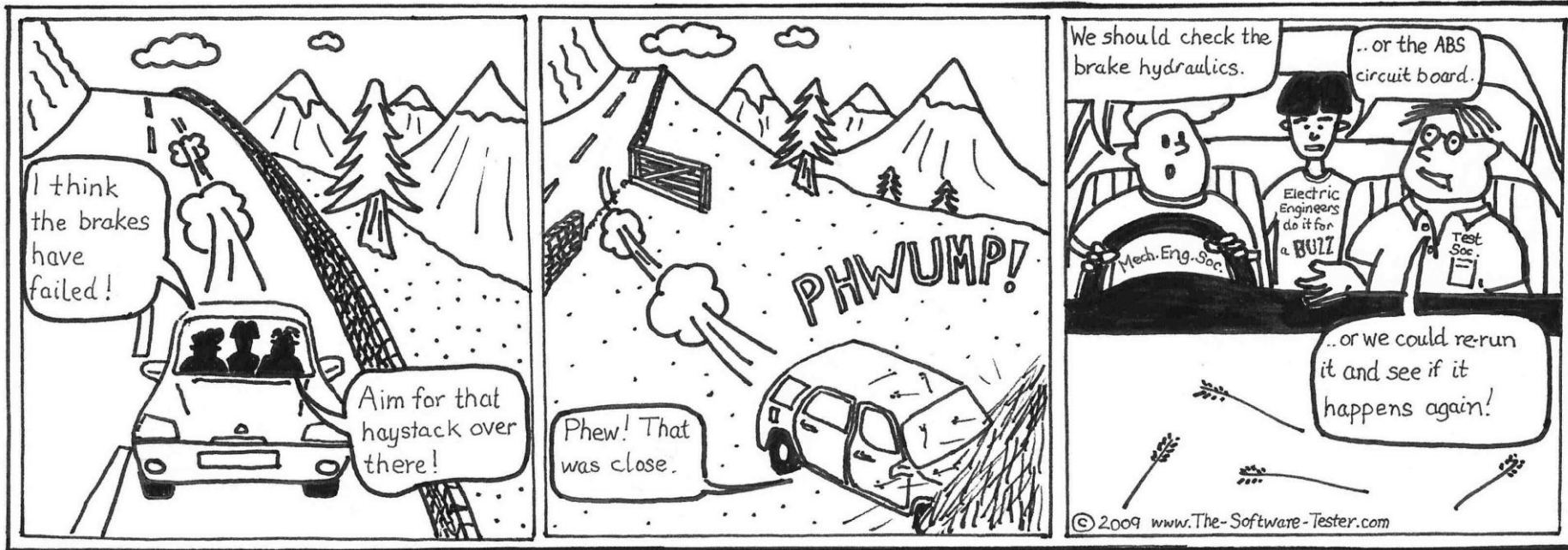
What we want to do today?

# Testing Career

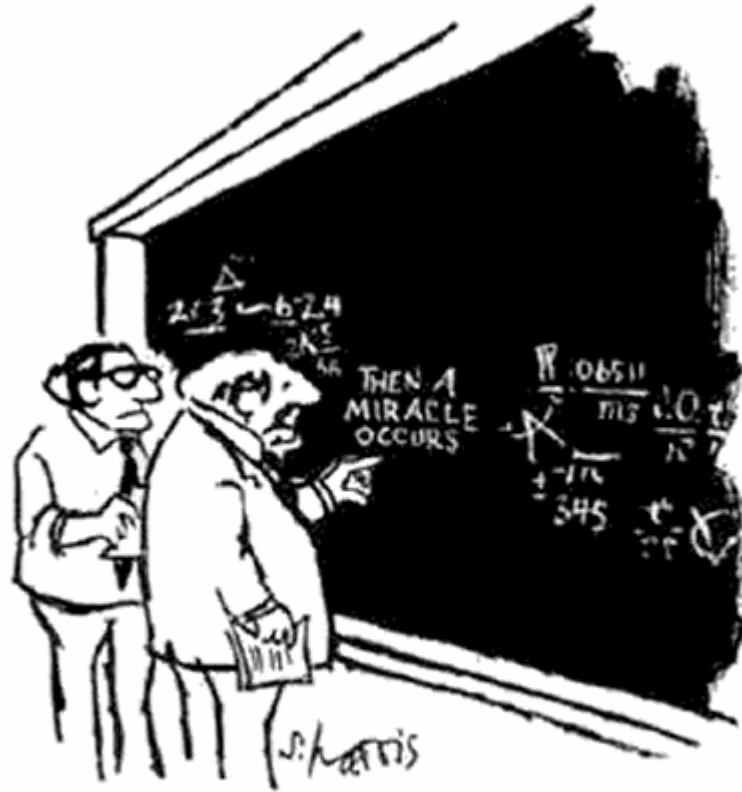


Andy Glover [cartoontester.blogspot.com](http://cartoontester.blogspot.com) copyright 2010

# What to Test?



# How to Test?



"I THINK YOU SHOULD BE MORE EXPLICIT  
HERE IN STEP TWO."

© 1985, 1979-80 | J. Morris

Distributed By Oxford-Engineering Ltd

# When to Stop Testing? (1/3)

## **“Quality Is in the Eye of the Beholder”**

Debanjan Mitra and Peter N. Golder  
Harvard Business Review, April 2007

## **“Quality is in the eye of the beholder: meeting users' requirements for Internet quality of service”**

Anna Bouch, Allan Kuchinsky, Nina Bhatti  
Proceedings of the ACM Conference on Human Factors in Computing  
Systems(2000) Volume: 2, Issue: 1, Publisher: ACM, Pages: 297-304

# When to Stop Testing? (2/3)

*for a piece of software/application so that the application could be considered to be "Quality Software"?*

*This is a fairly open ended question, but we did this on purpose. Whenever we have asked the question it has generated much debate and thought. We hope that the magazine will also engender much thought and debate when people read yours and other's responses.*

*We are not trying to define software quality, but are trying to collect ideas about it, and we believe that yours would be valuable.*

Below are the responses we got. Again, we are very grateful to the people who took the time and effort to write to us.

*Richard Bornet*

# When to Stop Testing? (3/3)



**"So, its fair to say you're very confident that you'll hit these testing milestones?"**

**Now.. Let's Welcome Our Panelists**