



# CS1632, Lecture 7: Smoke and Exploratory Testing

Bill Laboon




# Exploratory Testing

- We have developed a very formal manner of testing
  - Develop requirements
  - Write test plan
  - Create and check traceability matrix
  - Execute tests




# Exploratory Testing

- But we have assumed that we know the EXACT expected behavior, EXACTLY how to cause it, and it is necessary to DEFINE all of these behaviors
  - Works fine in some circumstances!
  - But not others!
- If I asked you to “test a poker program”, what would you do?

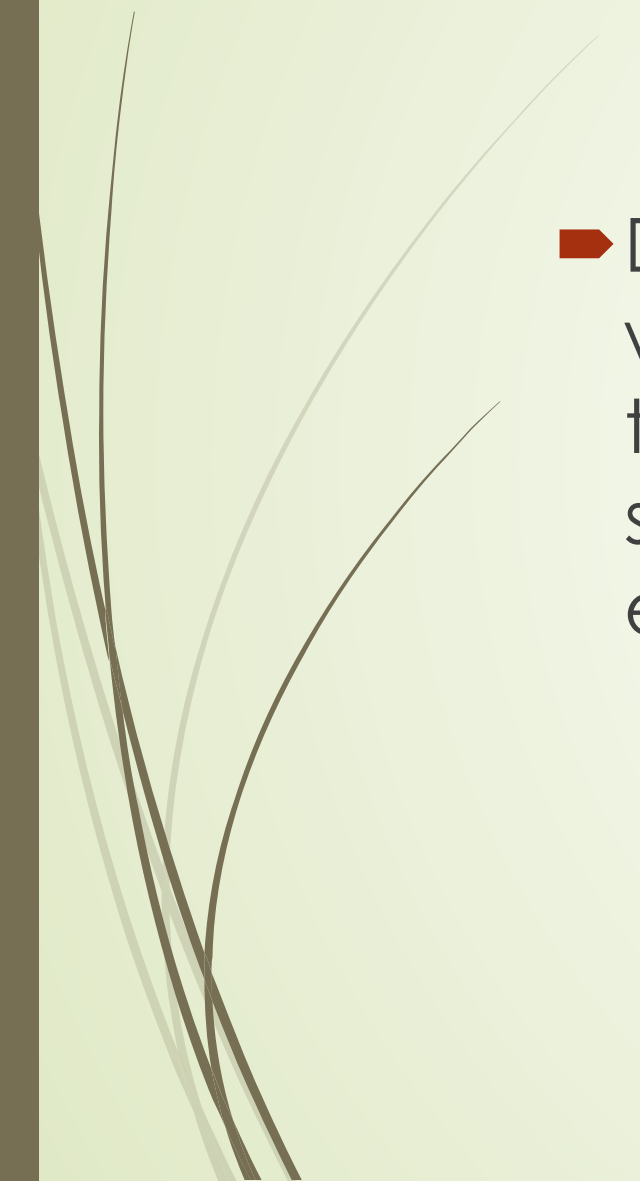


Sometimes, we don't know exactly what the expected behavior is! Why not?

- Subjective
  - Domain-specific
  - Uncertain of exact reproduction steps
  - Uncertain of interface
  - Unfamiliarity with general interaction
  - Implicit requirements
- 



# Exploratory Testing

- Definition: testing without a specific test plan, in which the goals are to both learn more about the system and inform the development of system by finding defects and possible enhancements
- 




## Sometimes called “*ad hoc*” testing

- Personally, I don't like this term
- It implies carelessness
- Less rigid != more careless
- Faith in the testers is required
  - To not go down blind alleys
  - To use their best judgment

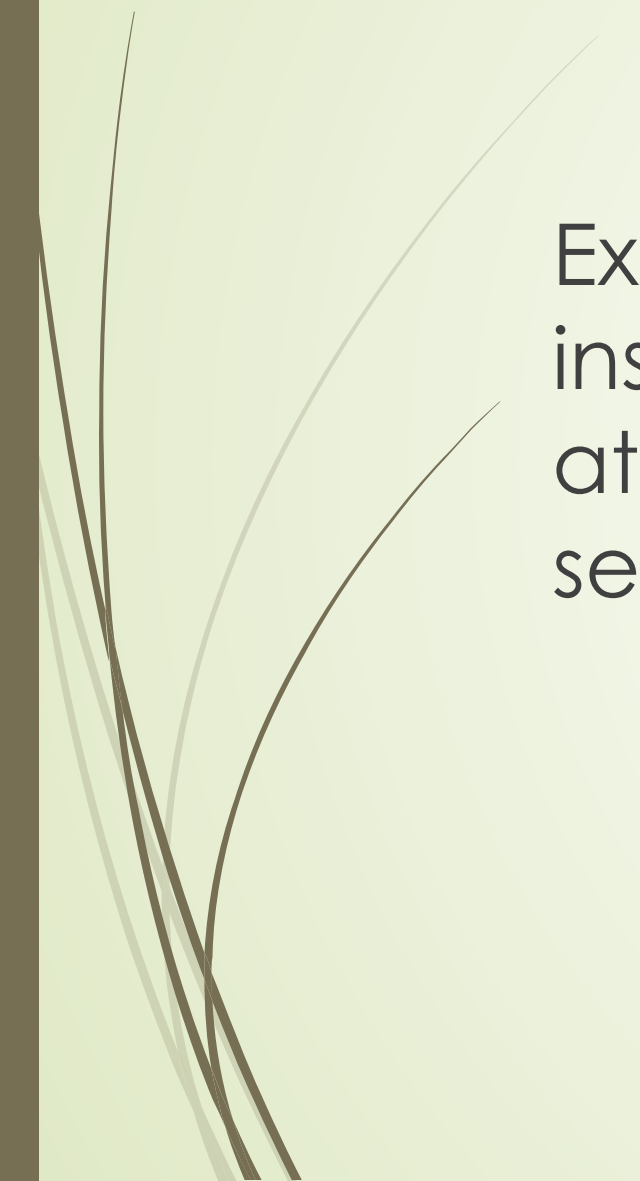


## How To Do It

1. Use your best judgment
2. If in doubt about next step, see Step 1.



# Faith in Testers




Exploratory testing has faith that you instinctively "know" that there's a defect, or at least that you know something doesn't seem quite right.






## Tips:

1. Try to accomplish important tasks
  2. Think of edge cases on the fly
  3. Try doing different things together
  4. If I were the programmer, what wouldn't I have thought of?
  5. Write down defects IMMEDIATELY
  6. You can keep track of your steps and write them down later as formal tests.
- 




# Benefits of Exploratory Testing

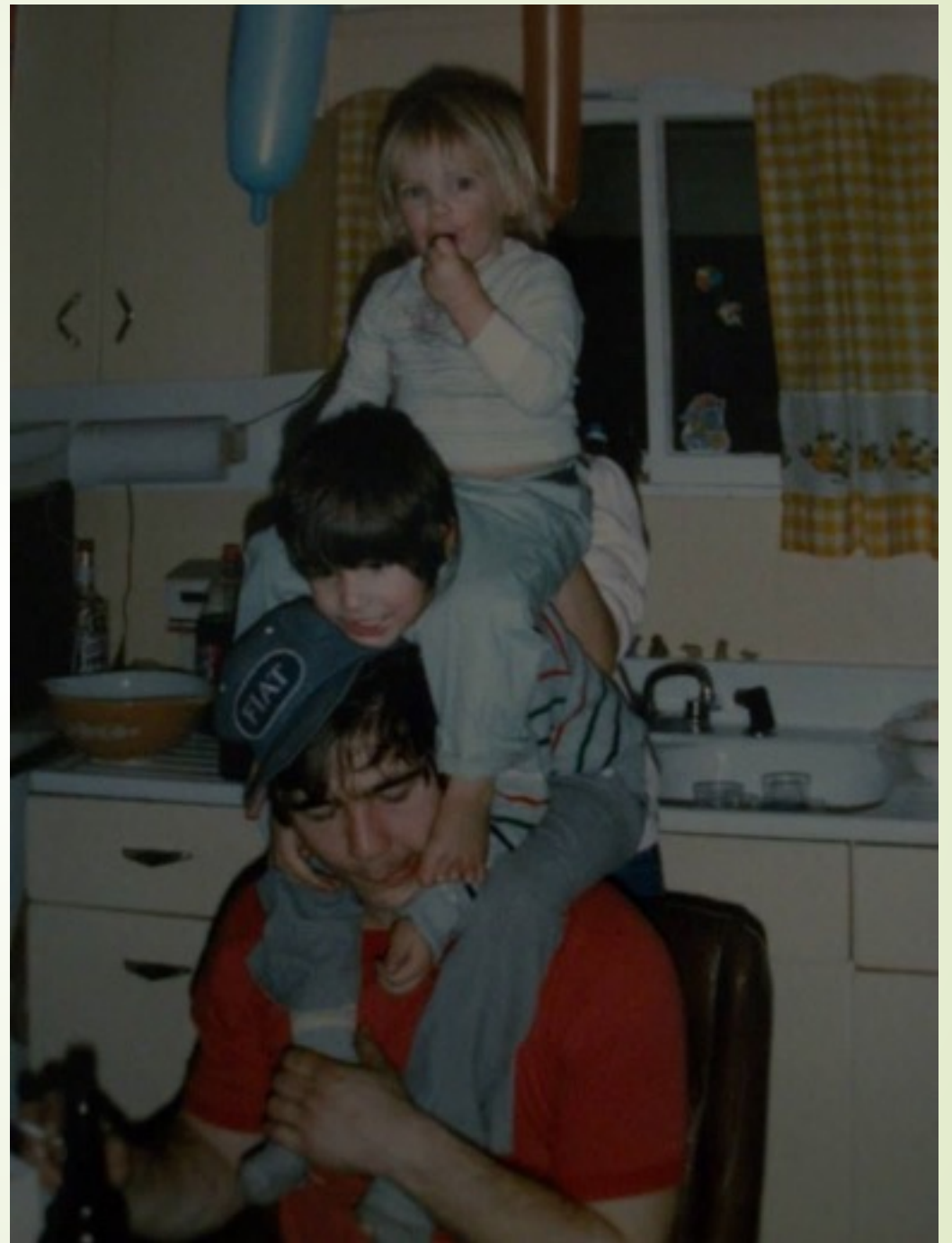
1. Fast
  2. Flexible
  3. Relies on testers' knowledge, and helps improve it
  4. Very easy to update!
- 



# Drawbacks to Exploratory Testing

- 
1. Unregulated
  2. Possibly unrepeatable
  3. Hard to say how much coverage there is
  4. Difficult to automate

# Smoke Testing





# Smoke Testing (plumbing)

- Send smoke down the pipes to find leaks BEFORE sending water or other fluids
- Why?
  - If there is a leak, much easier to clean up / find smoke
  - Won't waste effort
  - Won't cause further damage (high pressure water going through a hole means a bigger hole will be formed)




# Smoke Testing (software)

- ▶ Do some minimal testing to ensure that the system is, in fact, testable or ready to be released
- ▶ Why?
  - ▶ No need to test system that can't perform minimal acceptable functionality
  - ▶ Setting up test harnesses / installing software may be non-trivial
  - ▶ Avoid wasting testers' time

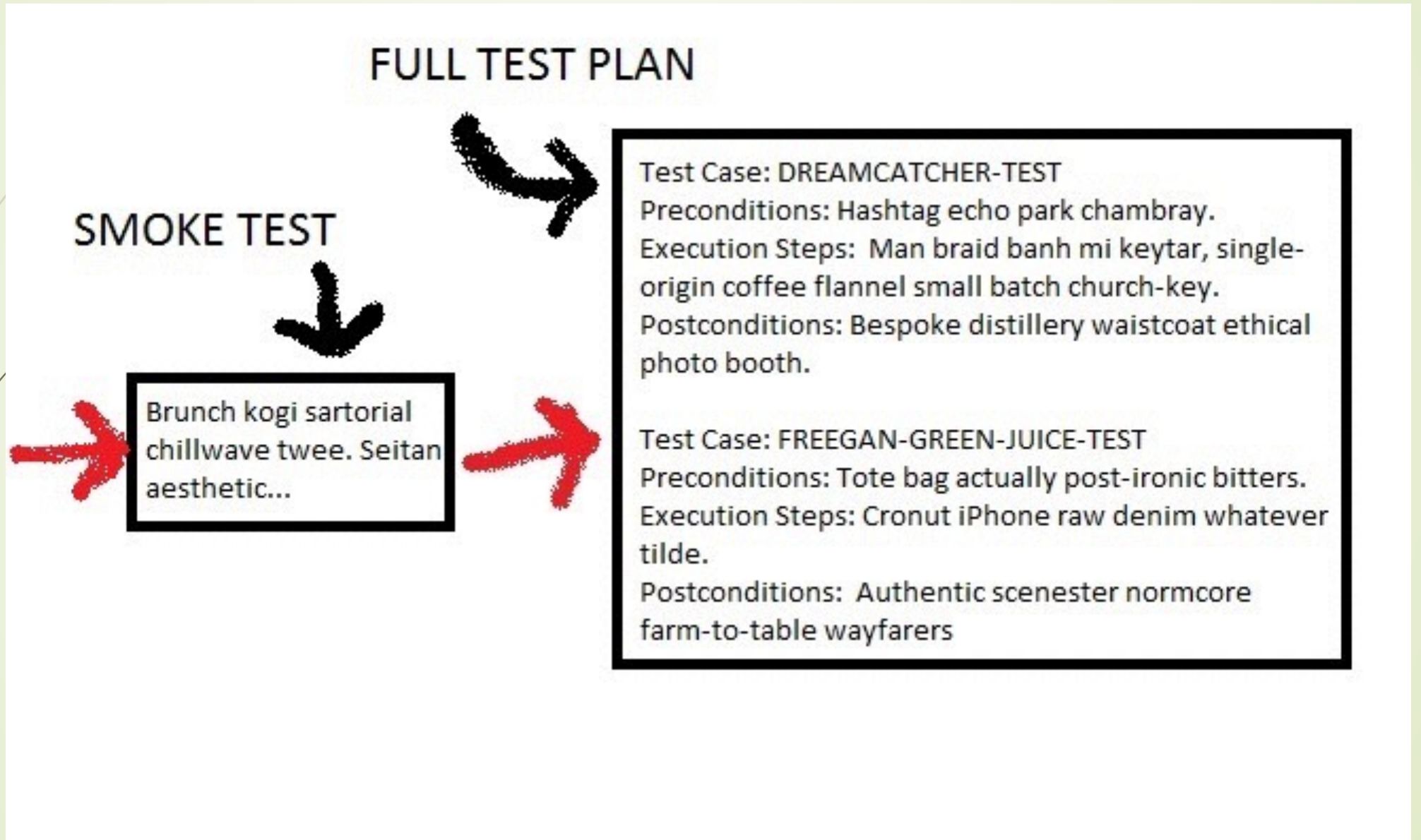


# Smoke Testing can be:

- **Scripted:** A few small but important test cases are run before the software is ready to be tested. These can be automated or manual.
  - **Unscripted:** An experienced tester does exploratory testing for a small amount of time to ensure that it meets minimum standards.
- 



# Smoke Testing is a GATEWAY







# Media Check

- ▶ A really, really basic smoke test
  - ▶ Can the CD be read?
  - ▶ Do files exist on server?
  - ▶ Etc.



# A Note on “Sanity Testing”

- Note: Some texts use the term “sanity testing” for “smoke testing”. I avoid this because:
  - It could be offensive
  - I think the parallel with smoke testing in plumbing is much more apt
- However, you may come across the term so I wanted to cover it