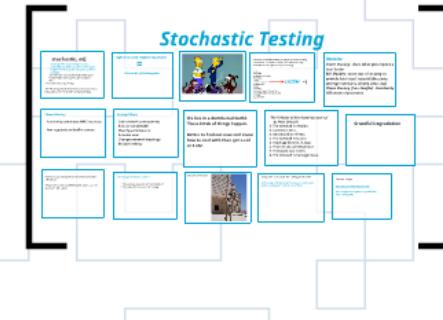
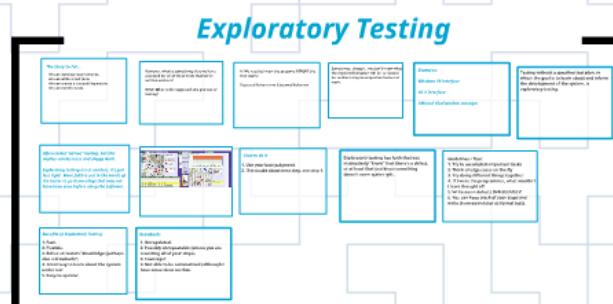


Let's Get Informal: Exploratory, Smoke and Stochastic Testing



Let's Get Informal: Exploratory, Smoke and Stochastic Testing



Exploratory Testing

The Story So Far...

We can develop requirements.
We can write a test plan.
We can create a traceability matrix.
We can run the tests.

However, what is something that we have assumed for all of these tests that we've written and run?

HINT: What is the supposed *sine qua non* of testing?

A: We need to know the outcome BEFORE the test starts!

Expected Behavior vs Observed Behavior

Sometimes, though... we don't know what the expected behavior will be, or should be, or there may be subjective factors at work.

Examples:

[Windows 10 Interface](#)

[OS X Interface](#)

[XMonad tiled window manager](#)

Often called "ad hoc" testing, but this implies carelessness and sloppy work.

Exploratory testing is not careless, it's just less rigid. More faith is put in the hands of the tester to go down alleys that may not have been seen before using the software.



How to do it

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

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.

Guidelines / 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 (perhaps also a drawback?).
4. Great way to learn about the system under test
5. Easy-to-update!

Drawbacks

1. Unregulated.
2. Possibly unrepeatable (unless you are recording all of your steps).
3. Coverage?
4. Not able to be automated (although I have some ideas on this).

The Story So Far...

We can develop requirements.

We can write a test plan.

We can create a traceability matrix.

We can run the tests.

However, what is something that we have assumed for all of these tests that we've written and run?

HINT: What is the supposed *sine qua non* of testing?

A: We need to know the outcome BEFORE the test starts!

Expected Behavior vs Observed Behavior

Sometimes, though... we don't know what the expected behavior will be, or should be, or there may be subjective factors at work.

Examples:

Windows 10 interface

OS X interface

XMonad tiled window manager

Testing without a specified test plan, in which the goal is to learn about and inform the development of the system, is *exploratory testing*.

Often called "ad hoc" testing, but this implies carelessness and sloppy work.

Exploratory testing is not careless, it's just less rigid. More faith is put in the hands of the tester to go down alleys that may not have been seen before using the software.

CoMotion - Main Desktop - Build Tue, 1 October 2002 18:00:14 EDT - User: intel - Server: 10.26.41.8

15:02 - intel

INTEL WS

Intel Map

Current View
North Sensor
South Sensor
Scout Pit
Pt Knox
Full View

intel UAV-1 Executing 0/02/02 14:59 to 0/02/02 17:59
Fly route shows as attached task w/ EO, IR/SAR package.

intel UAV-2 Executing 0/02/02 15:35 to 0/01/04 16:25
Fly route shows as attached task w/ EO, IR/SAR package orbiting twice over Green 27 Company

HECH
Pit Ped Activity
Infantry and trucks

TANK
Pit Black Destroyed
AMM 90s

Salute Report

Event ID	Event Description	Timestamp	Location	Activity	Orientation	Comments
10/02/0...	UAV 1	10/02/02 16:25	6 D-30, 8 trucks	Firing	North	
10/02/0...	D HQ-CRR	10/02/02 16:19	18 birds have	Firing	North	
10/02/0...	1st Sec. Scout	10/02/02 16:19	4 HMM hummers	Firing	East	
10/02/0...	D HQ-CRR	10/02/02 16:14	8mfs highgamma	Firing	West	
10/02/0...	3d Plat, A Co	10/02/02 16:09	1 MG vehicle	Moving	S.East	1 killed, one broke contact and moved south east
10/02/0...	3d Plat, A Co	10/02/02 15:47	20 pers., v 82mm	Firing	N.East	12 mort rounds impactDone vehicle hitRequest med vac for 3
10/02/0...	D HQ-CRR	10/02/02 15:04	UNKNOWN	UNKNOWN		12 Rounds
10/02/0...	Sensor 2	10/02/02 16:04	4 T-72	Moving	N.East	25ph
10/02/0...	Sensor 2	10/02/02 16:09	4 tanks	Moving	East	25ph
10/02/0...	4th Plat, A Co	10/02/02 16:43	Halted	East	3 T-72 and BTR/loop in support halted. Setting in off main road	
10/02/0...	UAV 2	10/02/02 16:42	10 T-72s, 4 BTR-40s	Moving	East	30 kph
10/02/0...	4th Plat, C Co	10/02/02 16:25	4 T-72	Destroyed	N.East	4 T-72s destroyed by arty/HM
10/02/0...	HUMINT	10/02/02 15:04	5 HW/HW w/12.7	UNKNOWN	South	ambushed our lead vehicle/Dran south
10/02/0...	UAV 2	10/02/02 16:14	50+ pers., mortars, HMM...	Halted	South	Apparent assy area

Frame Dispenser

- General Frames
 - +Sensors
 - +Recon
 - +Maneuver
 - +A Co
 - +B Co
 - +C Co
 - +27th Co
 - +Tubes
 - +Fires
 - +Tubes
 - +Motor Sec, A Co
 - +Motor Sec, B Co
 - +Motor Sec, C Co
 - +Motor Plot
 - +D Btry
- Tables
 - +Salute Report Table
 - +Scratch Salute Table
 - +Strike Table
 - +Scratch Strike Table
 - +Table
- Unit Frames
 - +Name
 - +Pit
 - +Ped
 - +Activity
 - +Infantry and trucks
- Reference Frames
 - +Overview Map
 - +Intelligence Order of Battle
 - +Green Order of Battle
 - +Harmless Order of Battle
 - +Red Order of Battle
- Viewers / Editor Frames
 - +Salute Viewer / Editor
 - +Strike Viewer / Editor
 - +Salute Sensor Chart
 - +Salute Assertion Chart

PLAYERS

CO WS FSC WS ACO WS RCO WS CCO WS BDA

Master Schedule

Master Schedule

10/02/02 16:17

CCDR#1 - Main Eff/Type
Co Red Activity
Determine Main Effects. Indicators: T-72, BM-21, Recon

CCDR#2 - Guerrilla Type
Co Red Activity
Determine guerrilla activities and intent with emphasis on supporting red force attack.

TF Frage
TF defends area around airfield IOT prevent enemy forces from controlling area and moving north to capital. Companies occupy designated positions on CO map. Priority of fires to A Co.

oo Oct 2, 2002 9:30 AM AF

How to do it

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

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.

Guidelines / 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 (perhaps also a drawback?)**
- 4. Great way to learn about the system under test**
- 5. Easy-to-update!**

Drawbacks

- 1. Unregulated.**
- 2. Possibly unrepeatable (unless you are recording all of your steps).**
- 3. Coverage?**
- 4. Not able to be automatized (although I have some ideas on this).**

Smoke Testing



Smoke testing (PLUMBING): send smoke down the pipes to find leaks BEFORE sending water or other fluids.

WHY?
Much easier to clean up smoke than water.

Won't waste effort - hooking up to a water main is non-trivial.
Won't cause further damage (high-pressure water going through a hole => bigger hole).

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 functionality.

Setting up test harnesses etc. is non-trivial.

May waste time going down blind alleys.

Smoke testing can be:

1. Scripted : There are a few small but important test cases (taking an hour or two to execute, at most) which are run prior to the software being released to the greater team.
2. Unscripted: An experienced tester does exploratory or ad hoc testing for an hour or so.

Keep in mind:
Smoke testing is an ADDITION to traditional software testing. It is a GATEWAY to further testing or release.



Media Check

A really, really basic smoke test:
Can the CD be read?
Do files exist on the server?
etc.

NB: Some texts use the term "sanity" testing for "smoke testing". I try to avoid this term as it may be offensive.

Smoke testing is "part of a complete breakfast."



Smoke testing (PLUMBING): send smoke down the pipes to find leaks BEFORE sending water or other fluids.

WHY?

Much easier to clean up smoke than water.

Won't waste effort - hooking up to a water main is non-trivial.

Won't cause further damage (high-pressure water going through a hole -> bigger hole).

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 functionality.

Setting up test harnesses etc. is non-trivial.

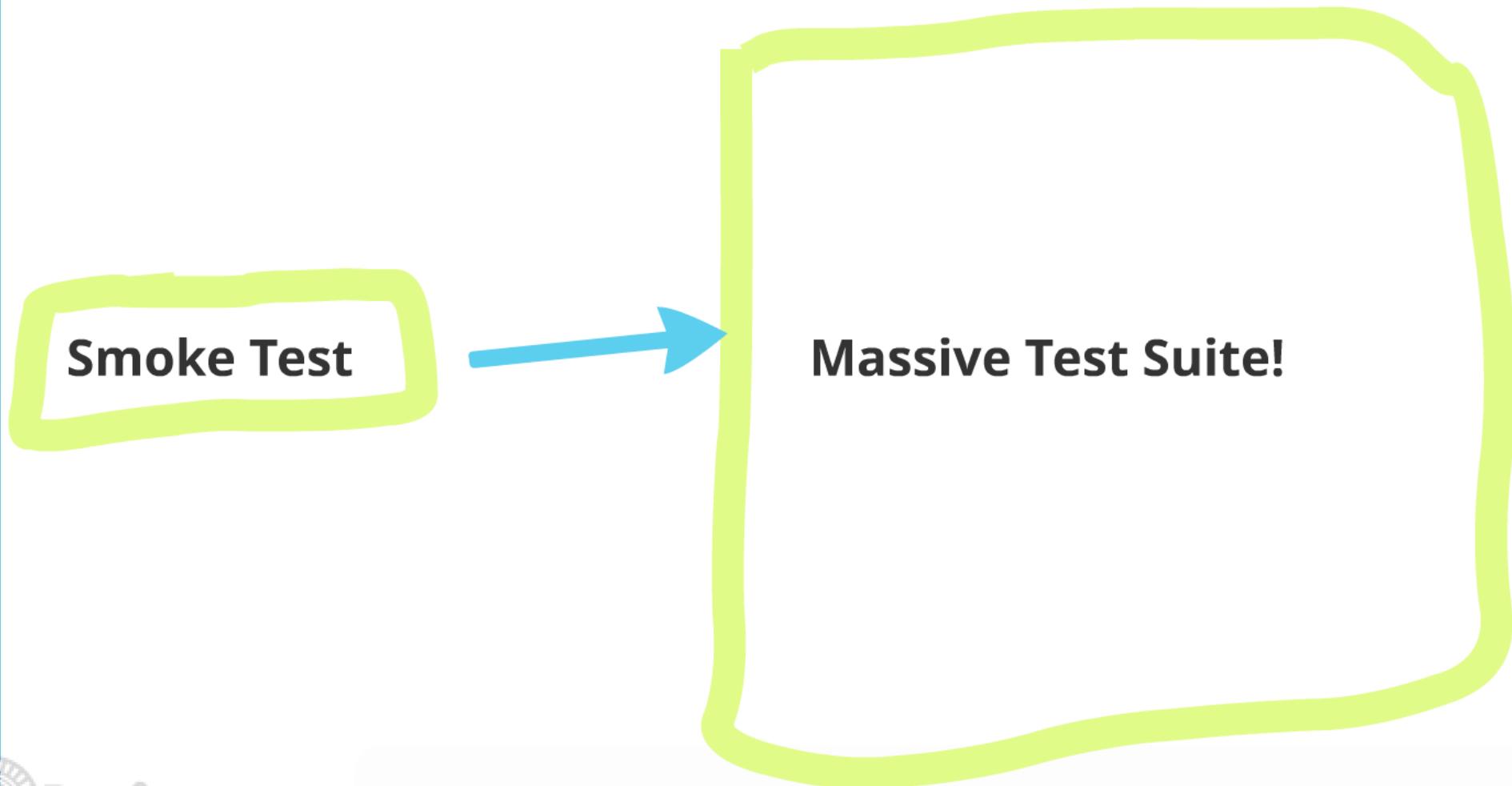
May waste time going down blind alleys.

Smoke testing can be:

- 1. Scripted :** There are a few small but important test cases (taking an hour or two to execute, at most) which are run prior to the software being released to the greater team.
- 2. Unscripted:** An experienced tester does exploratory or ad hoc testing for an hour or so.

Keep in mind:

Smoke testing is an ADDITION to traditional software testing. It is a GATEWAY to further testing or release.



Media Check

A really, really basic smoke test:

Can the CD be read?

Do files exist on the server?

etc.

NB: Some texts use the term "sanity" testing for "smoke testing". I try to avoid this term as it may be offensive.

Smoke testing is "part of a complete breakfast."

Stochastic Testing

stochastic, adj.

randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.
mid 17th cent.: from Greek stokhastikos, from stokheasthai 'aim at, guess,' from stokhos 'aim.'

Also called "monkey testing".

PROTTIP: Using words with Greek roots sounds more impressive than words named after primates.

Infinite monkey + Infinite typewriters
=
the works of Shakespeare



Think of stochastic testing as property based testing, with few or no limits on input, and often the only invariant is "the system keeps running!"

```
2 + 2 = 4
lambda(x)
  (greedy hands emoji)
  newjewels3
  > 999
  {"car": "values"}
  ()@()@()@()@()
  fehish
  @nigabisa
  let main () => printf ("Ineson");;
  ALL LIVING CREATURES DIE ALONE
  e++@JS
  N@P@P@P@E@U@S@
```

Variants:

Smart Monkey - does what you expect a user to do

Evil Monkey - Goes out of its way to provide bad input (executable code, strange numbers, binary data, etc)

Chaos Monkey (from Netflix) - Randomly kill servers/processes

Chaos Monkey

Randomly terminates AWS instances
Run regularly on Netflix servers

Related Tools

Cut network connectivity
Reduce bandwidth
Modify permissions
Remove user
Change network topology
Induce latency

We live in a distributed world.
These kinds of things happen.

Better to find out now and know how to deal with than get a call at 3 AM.

"The Fallacies of Distributed Computing"

- by Peter Deutch
1. The network is reliable.
 2. Latency is zero.
 3. Bandwidth is infinite.
 4. The network is secure.
 5. Topology doesn't change.
 6. There is one administrator.
 7. Transport cost is zero.
 8. The network is homogeneous.

Graceful Degradation

"The best way to avoid failure is to fail constantly." - Jeff Atwood

"If you want to make a difficult task easier, do it all the time." - Bill Laboon

The more you deal with a problem:

1. The more you know HOW to deal with it
2. The easier it is to automate it away

Example: Generators



Computers have moved from being pets to cattle.

Testing large, distributed systems means treating your systems as fungible "units of computation".

Example: Google

<http://googletesting.blogspot.com/>

"How Google Tests Software" by Whittaker, Arbon, and Carolla

stochastic, adj.

randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

*mid 17th cent.: from Greek *stokhastikos*, from *stokhazesthai* ‘aim at, guess,’ from *stokhos* ‘aim.’*

Also called "monkey testing".

PROTIP: Using words with Greek roots sounds more impressive than words named after primates.

Infinite monkey + Infinite typewriters



the works of Shakespeare



**Think of stochastic testing as property based testing,
with few or no limits on input, and often the only
invariant is "the system keeps running!"**

2 + 2 = 4

DOWAGER

(praying hands emoji)

fewjwe8r3

> 9000

{ "attr" : "value" }

{}){{#U(d))))))}))

fehioe

0x98AB654

int main () { printf ("meow"); }

ALL LIVING CREATURES DIE ALONE

e=++)@\$

N((#HIFHEUIE

 **SYSTEM** =)

Variants:

Smart Monkey - does what you expect a user to do

Evil Monkey - Goes out of its way to provide bad input (executable code, strange numbers, binary data, etc)

Chaos Monkey (from Netflix) - Randomly kill servers/processes

Chaos Monkey

Randomly terminates AWS instances

Run regularly on Netflix servers

Related Tools

Cut network connectivity

Reduce bandwidth

Modify permissions

Remove user

Change network topology

Induce latency

**We live in a distributed world.
These kinds of things happen.**

**Better to find out now and know
how to deal with than get a call
at 3 AM.**

"The Fallacies of Distributed Computing"

-by Peter Deutsch

- 1. The network is reliable.**
- 2. Latency is zero.**
- 3. Bandwidth is infinite.**
- 4. The network is secure.**
- 5. Topology doesn't change.**
- 6. There is one administrator.**
- 7. Transport cost is zero.**
- 8. The network is homogeneous.**

Graceful Degradation

**"The best way to avoid failure is to fail constantly." -
Jeff Atwood**

**"If you want to make a difficult task easier, do it all
the time." -Bill Laboon**

The more you deal with a problem:

1. The more you know HOW to deal with it
2. The easier it is to automate it away

Example: Generators



Computers have moved from being *pets* to *cattle*.

Testing large, distributed systems means treating your systems as fungible "units of computation".

Example: Google

<http://googletesting.blogspot.com/>

*"How Google Tests Software" by Whittaker,
Arbon, and Carollo*

Let's Get Informal: Exploratory, Smoke and Stochastic Testing

