Announcements

- Project 1 grades out
  - Mean: 23.4/25

- Assignment 3 due today

- Assignment 4 releasing today

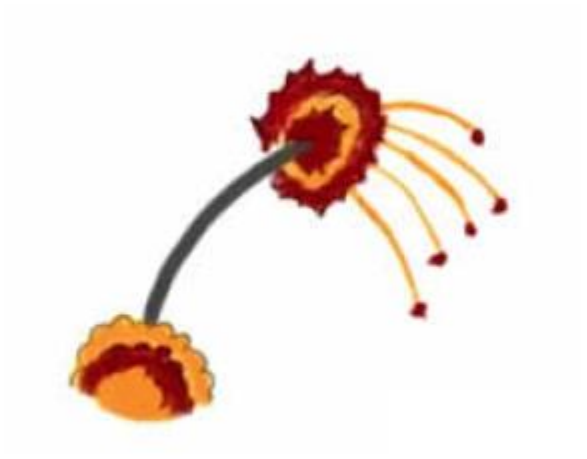
CS3300 Introduction to Software Engineering

Lecture 15: Software Testing

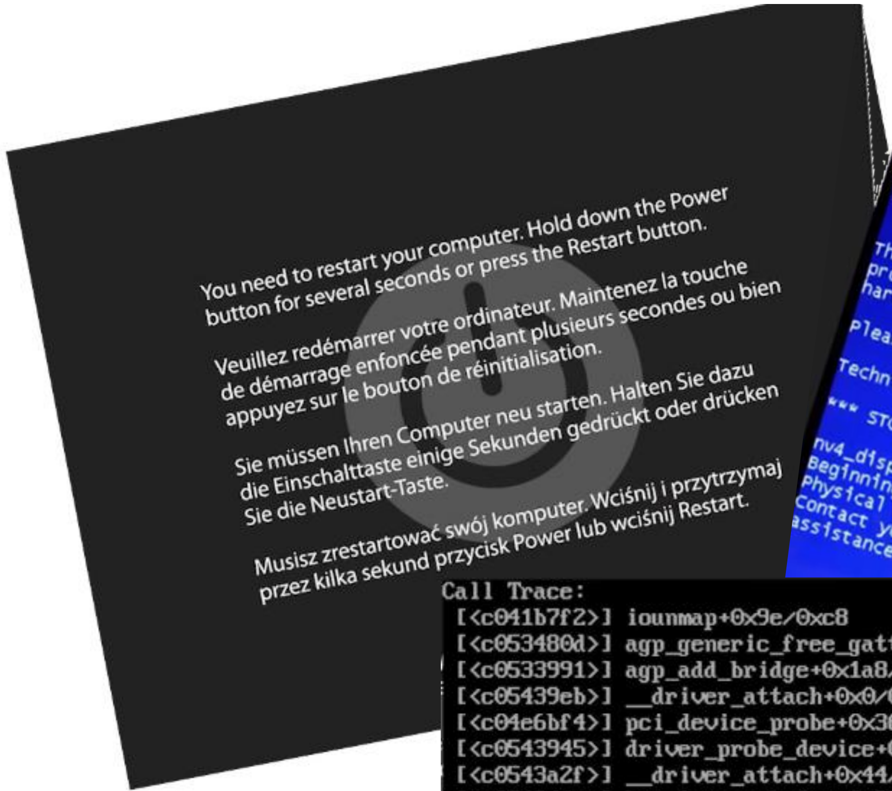
Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

Some Examples…



Ariana 5 Failure:
https://www.youtube.com/watch?v=gp_D8r-2hwk

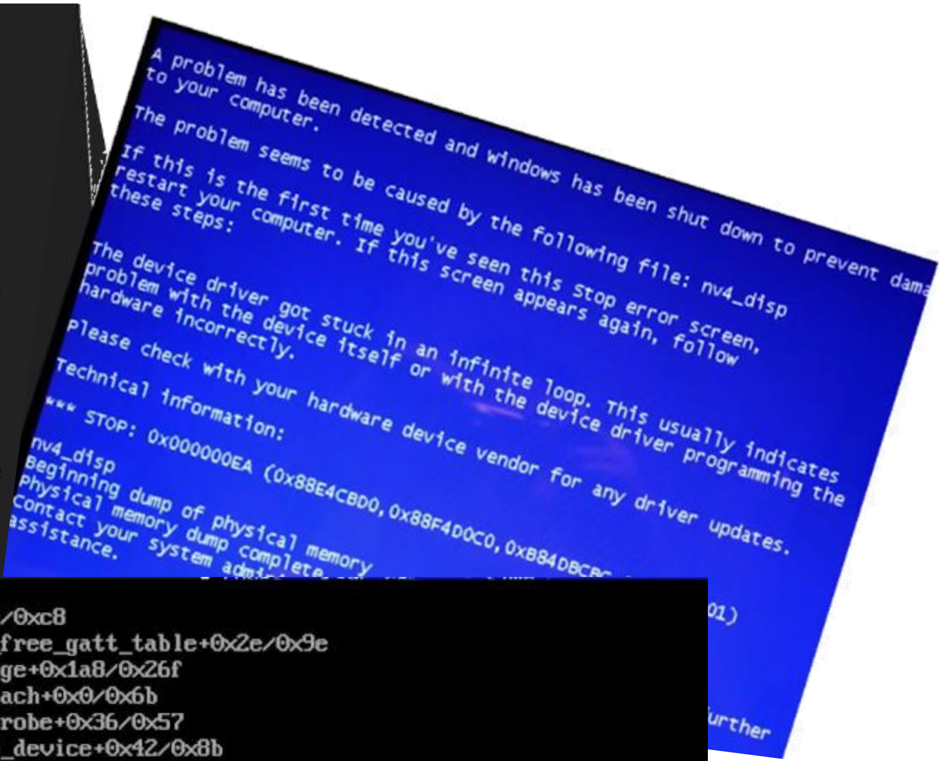


You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

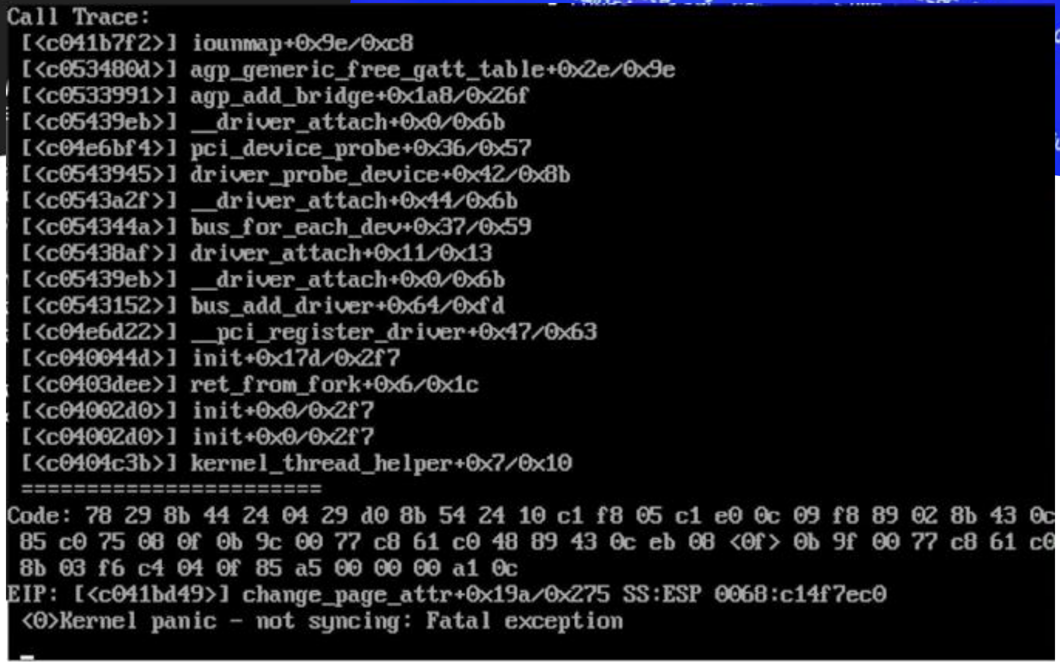
Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

Musisz zrestartować swój komputer. Wciśnij i przytrzymaj przez kilka sekund przycisk Power lub wciśnij Restart.



```
A problem has been detected and windows has been shut down to prevent dama
to your computer.
The problem seems to be caused by the following file: nv4_disp
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
The device driver got stuck in an infinite loop. This usually indicates
problem with the device itself or with the device driver programming the
hardware incorrectly.
Please check with your hardware device vendor for any driver updates.
Technical information:
*** STOP: 0x000000EA (0x88E4CBD0,0x88F4D0C0,0xB84DBCB
nv4_disp
Beginning dump of physical memory
Physical memory dump complete
Contact your system admi
assistance.
01)
urther
```



```
Call Trace:
 [<c041b7f2>] iounmap+0x9e/0xc8
 [<c053480d>] agp_generic_free_gatt_table+0x2e/0x9e
 [<c0533991>] agp_add_bridge+0x1a8/0x26f
 [<c05439eb>] __driver_attach+0x0/0x6b
 [<c04e6bf4>] pci_device_probe+0x36/0x57
 [<c0543945>] driver_probe_device+0x42/0x8b
 [<c0543a2f>] __driver_attach+0x44/0x6b
 [<c054344a>] bus_for_each_dev+0x37/0x59
 [<c05438af>] driver_attach+0x11/0x13
 [<c05439eb>] __driver_attach+0x0/0x6b
 [<c0543152>] bus_add_driver+0x64/0xfd
 [<c04e6d22>] __pci_register_driver+0x47/0x63
 [<c040044d>] init+0x17d/0x2f7
 [<c0403dee>] ret_from_fork+0x6/0x1c
 [<c04002d0>] init+0x0/0x2f7
 [<c04002d0>] init+0x0/0x2f7
 [<c0404c3b>] kernel_thread_helper+0x7/0x10
 =======================
Code: 78 29 8b 44 24 04 29 d0 8b 54 24 10 c1 f8 05 c1 e0 0c 09 f8 89 02 8b 43 0c
 85 c0 75 08 0f 0b 9c 00 77 c8 61 c0 48 89 43 0c eb 08 <0f> 0b 9f 00 77 c8 61 c0
 8b 03 f6 c4 04 0f 85 a5 00 00 00 a1 0c
EIP: [<c041bd49>] change_page_attr+0x19a/0x275 SS:ESP 0068:c14f7ec0
 <0>Kernel panic - not syncing: Fatal exception
_
```

Slide adapted from Alessandro Orso

Software is Buggy!

- Cost of bugs: $ 60 B/year
- On average, 1-5 errors per 1KLOC
- Windows 10
  - 50M LOC
  - 63,000 known bugs at the time of release
  - 2 per 1,000 lines

- For mass market software 100% correct SW development is infeasible, but
- We must verify the SW as much as possible

Failure, Fault, Error

Failure: Observable incorrect behavior of a program. Conceptually related to the behavior of the program, rather than its code.

Fault (bug): Related to the code. Necessary (not sufficient!) condition for the occurrence of a failure.

Error: Cause of a fault. Usually a human error (conceptual, typo, etc.)

Failure, Fault, Error: Example

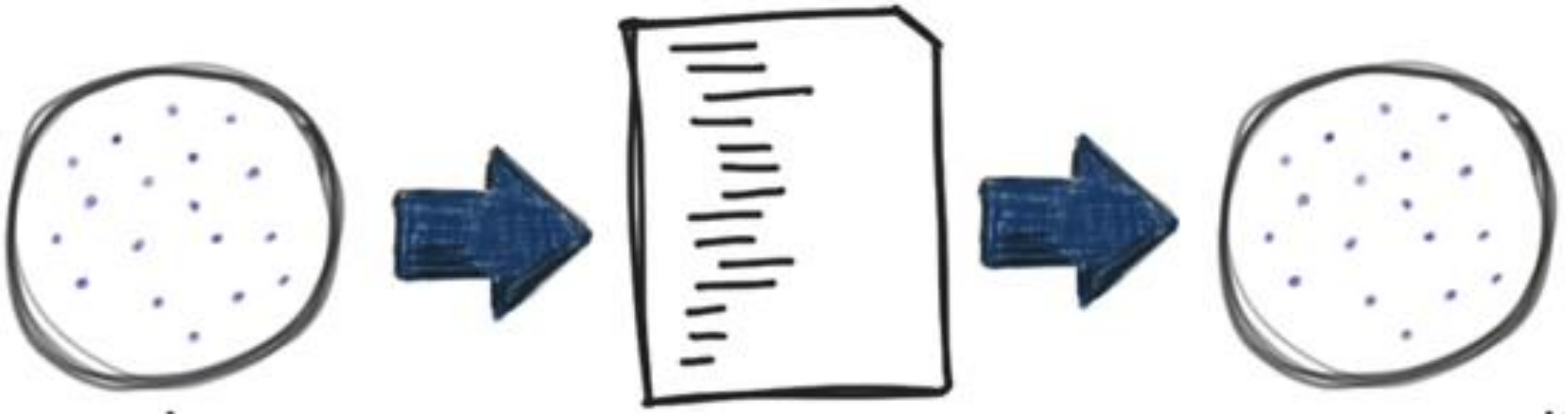
```
1.    double doubleValue(int param) {
2.        double result;
3.        result = (double) param * param;
4.        return(result);
5.    }
```

A call to double(3) returns 9. What is this?	The result 9 is a failure- it is an observable behavior
Where is the fault?	Line 3
What is the error that caused the fault?	N/A. Maybe typo, erroneous copy paste, or conceptual. Only the developer knows.

Approaches to Verification

- **Testing** (dynamic verification): exercising software to try and generate failures
- **Static analysis**: identify (specific) problems statically, that is, considering all possible executions
- **Inspections/reviews/walkthroughs**: systematic group review of program text to detect faults
- **Formal verification** (proof of correctness): proving that the program implements the program specification

Testing



Input Domain D

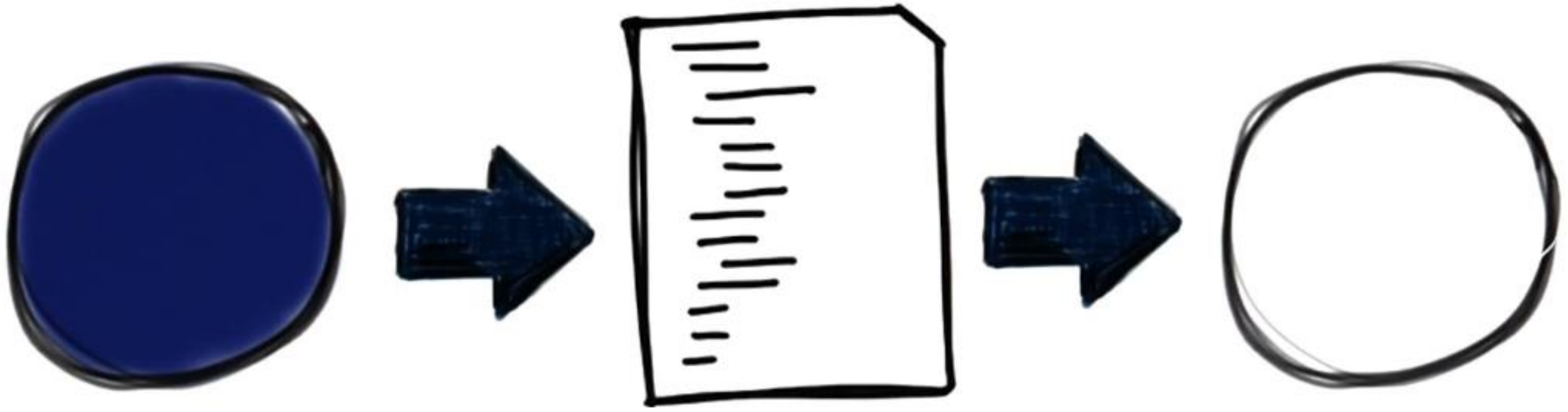
Software

Output Domain O

Test Case: {i ∈ D, o ∈ O}

Test Suite: A set of Test Cases

Static Verification



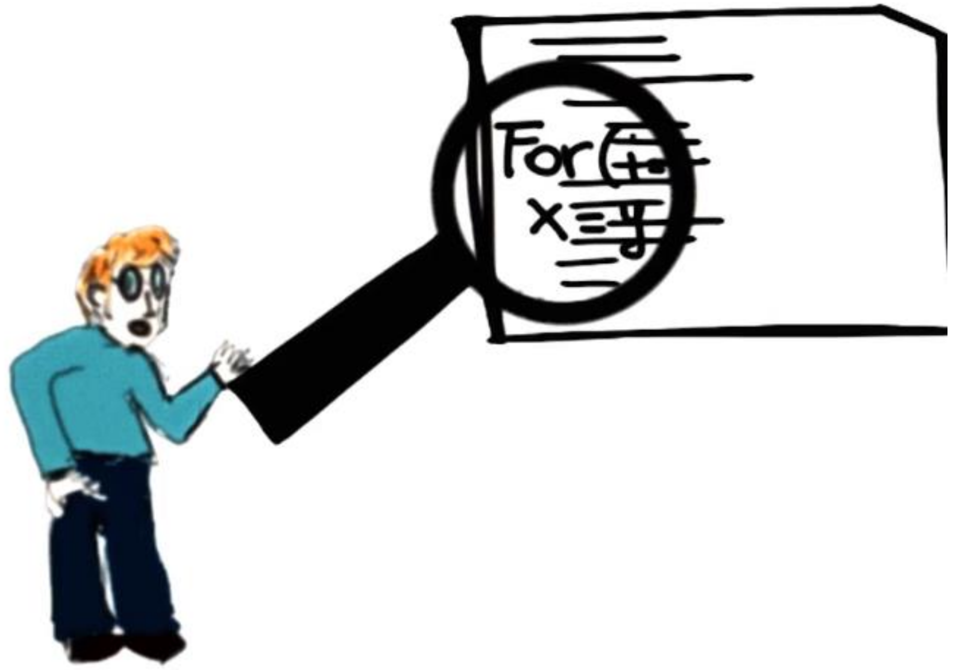
Input Domain D

Software

Output Domain O

Considers all possible inputs
(execution/behaviors)

Inspections/Reviews/Walkthroughs



Human intensive activity
Manual
Group activity
Inspect defects in the artifacts

Formal Proof (Of correctness)

Given a formal specification, checks that the code corresponds to such specification

Sophisticated mathematical analysis

Comparison among the 4 techniques

	PROS	CONS
Testing	No False Positives	Highly Incomplete
Static Verification	Considers all program behaviors, Complete	False Positives, Expensive
Inspections	Systematic, Thorough	Informal, Subjective
Formal Proofs of Correctness	Strong Guarantees	Complex, Expensive to build/prove a mathematical basis

Today, Quality Assurance (Verification) is mostly Testing

“50% of my company employees are testers, and the rest spend 50% of their time testing”. Who said that?

[ ] Mark Zuckerberg

[ ] Steve Jobs

[ ] Henry Ford

[✓] Bill Gates

[ ] Frank Gehry

What is Testing?

Testing == To execute a program with a sample of the input data

- Dynamic technique: program must be executed
- Optimistic approximation:
  - The program under test is exercised with a (very small) subset of all the possible input data
  - We **assume** that the behavior with any other input is consistent with the behavior shown for the selected subset of input data

Testing Techniques

There are several techniques

- Different processes
- Different artifacts
- Different approaches

There are no perfect techniques

- Testing is a best-effort activity

There is no best technique

- Different contexts
- Complementary strengths and weaknesses
- Trade-offs

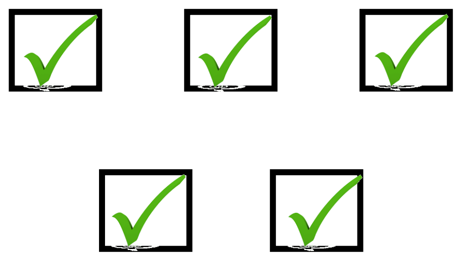
Successful Tests

"a test is successful if the program fails"

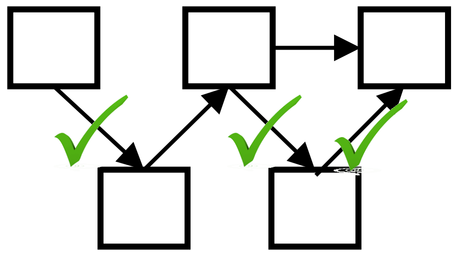
-Goodenough and Gerhart (1985). "Towards a Theory of Test data selection". *IEEE Transactions of Software Engineering*, Jan 1985

Testing Granularity Levels

Unit Testing

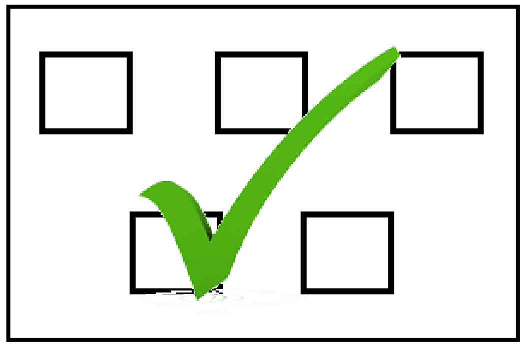


Integration



Big Bang

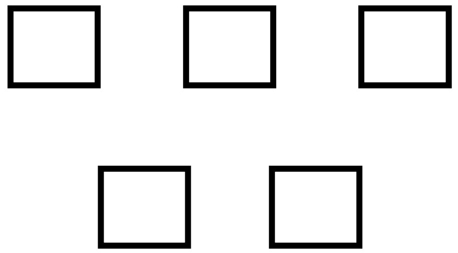
System



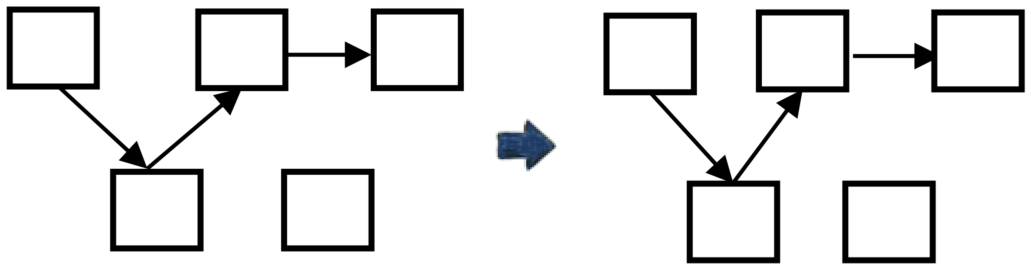
Functional/Non-functional

Customer

Acceptance Testing

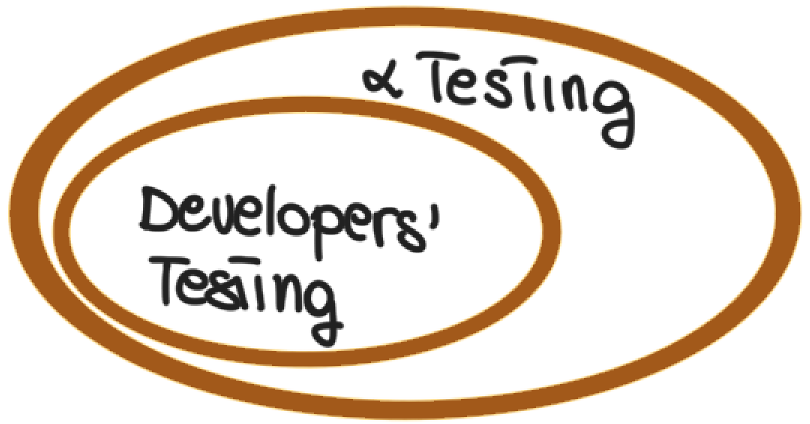


Regression Testing

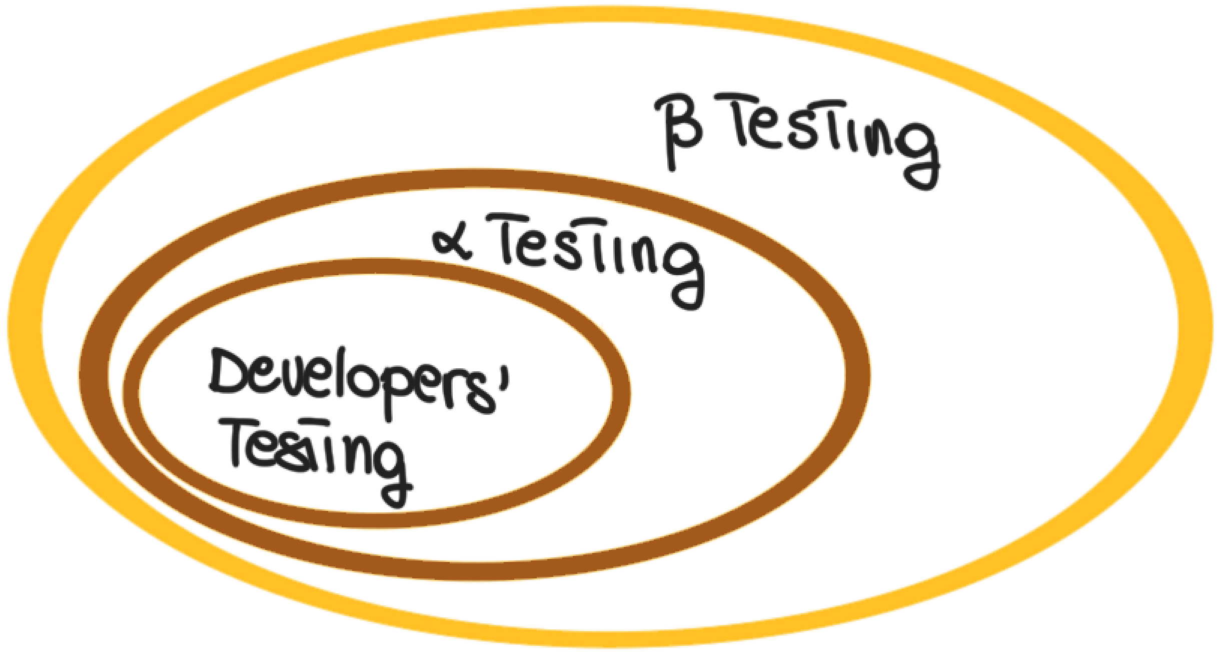


Testing Types

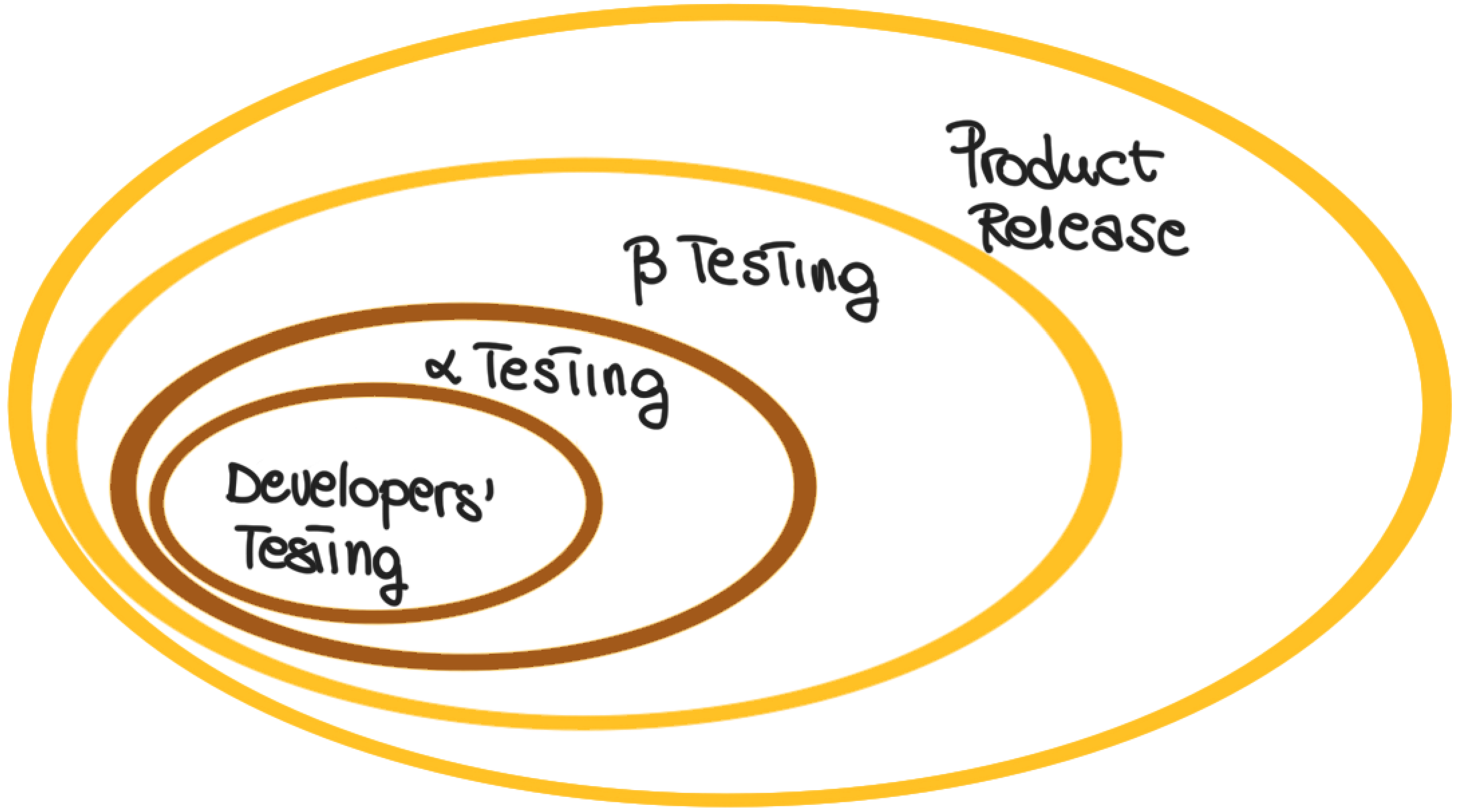
Testing Types



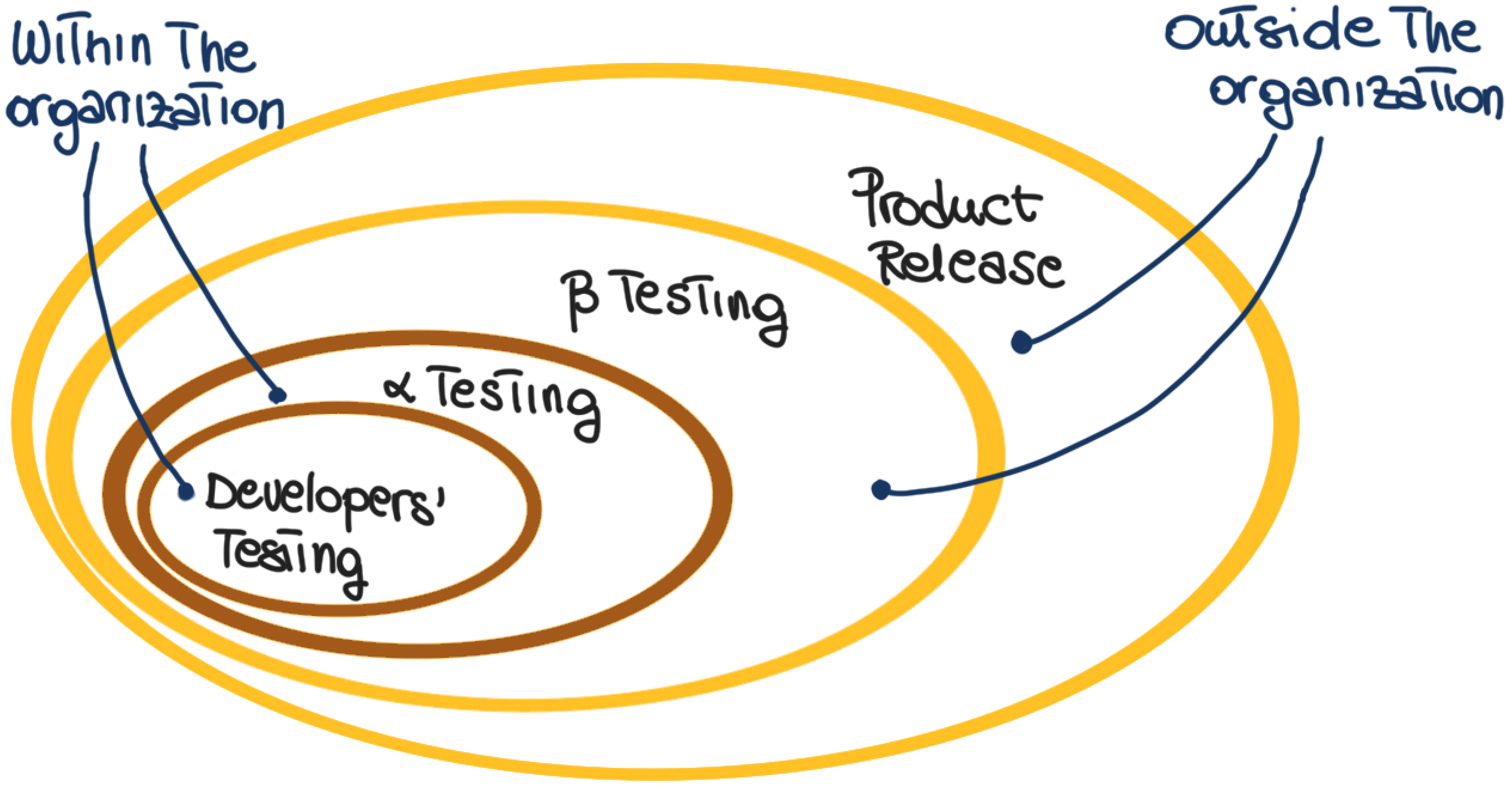
Testing Types



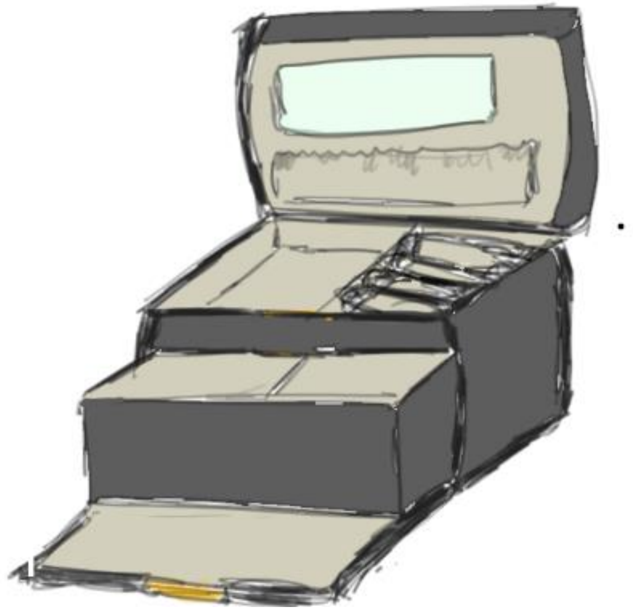
Testing Types



Testing Types



Testing Types



BLACK BOX TESTING

- Based on a description of the software (specification)
- Cover as much specified behavior as possible
- Cannot reveal errors due to implementation details

WHITE BOX TESTING

- Based on the code
- Cover as much coded behavior as possible
- Cannot reveal errors due to missing paths

Black-Box Testing Example

Specification: Inputs an integer and prints it

```
void printNumBytes (param){
     if (param < 1024) printf(“%d”, param);
    else  printf(‘%d KB” ,  param/124);
}
```

White-Box Testing Example

Specification: inputs an integer param and returns half of its value if even, its value otherwise

```
int  fun(int param){
  int result;
  result = param/2;
  return result;
}
```