SENG 275

SOFTWARE TESTING

DR. NAVNEET KAUR POPLI

DEPT. OF ELECTRICAL AND COMPUTER ENGINEERING

University of Victoria

FAULT / FAILURE / ERROR / BUG / DEFECT

VERIFICATION / VALIDATION

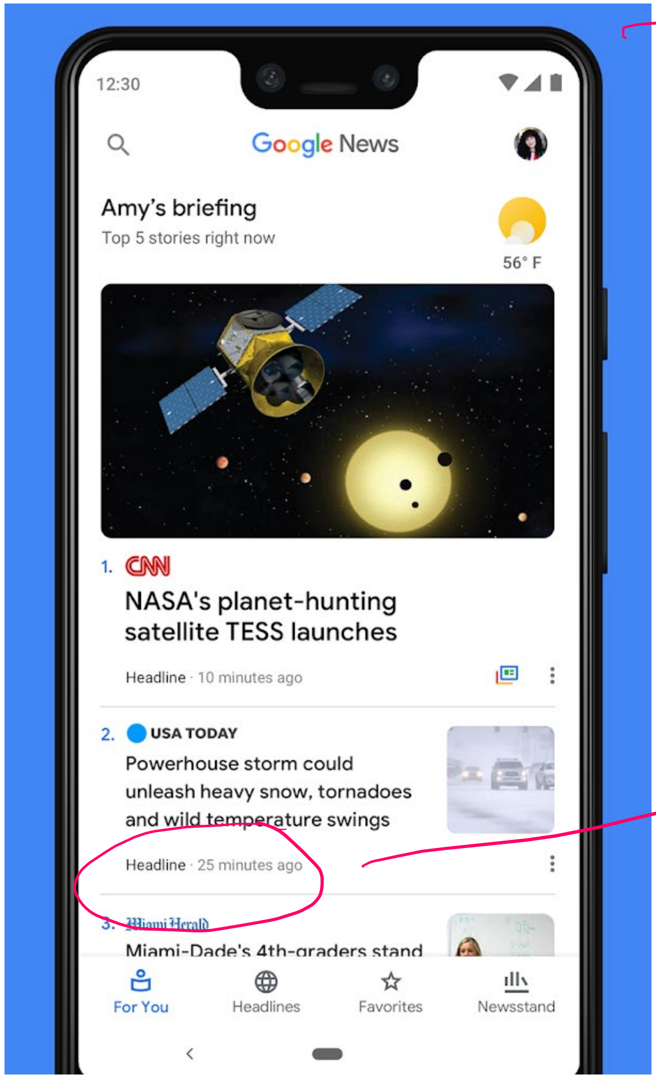
FAULT/FAILURE/ERROR/BUG/DEFECT

VERIFICATION/VALIDATION

University of Victoria

Failure, Fault and Error

- **Failure** is an OBSERVABLE INCORRECT BEHAVIOUR of the software.
- **Fault** is INCORRECT CODE.
- **Error** is CAUSE OF FAULT.
- If Failures become visible to the end user, it may be disastrous to the application.
- An example of a failure is a mobile app that stops working, or a news website that starts to show yesterday's news on its front page.
- The software system did something it was not supposed to do.
- Errors may be caused if specifications are incorrectly interpreted by a developer.



12:30

Google News

Amy's briefing

Top 5 stories right now

56° F

1. CNN

NASA's planet-hunting satellite TESS launches

Headline · 10 minutes ago

2. USA TODAY

Powerhouse storm could unleash heavy snow, tornadoes and wild temperature swings

Headline · 25 minutes ago

3. Miami Herald

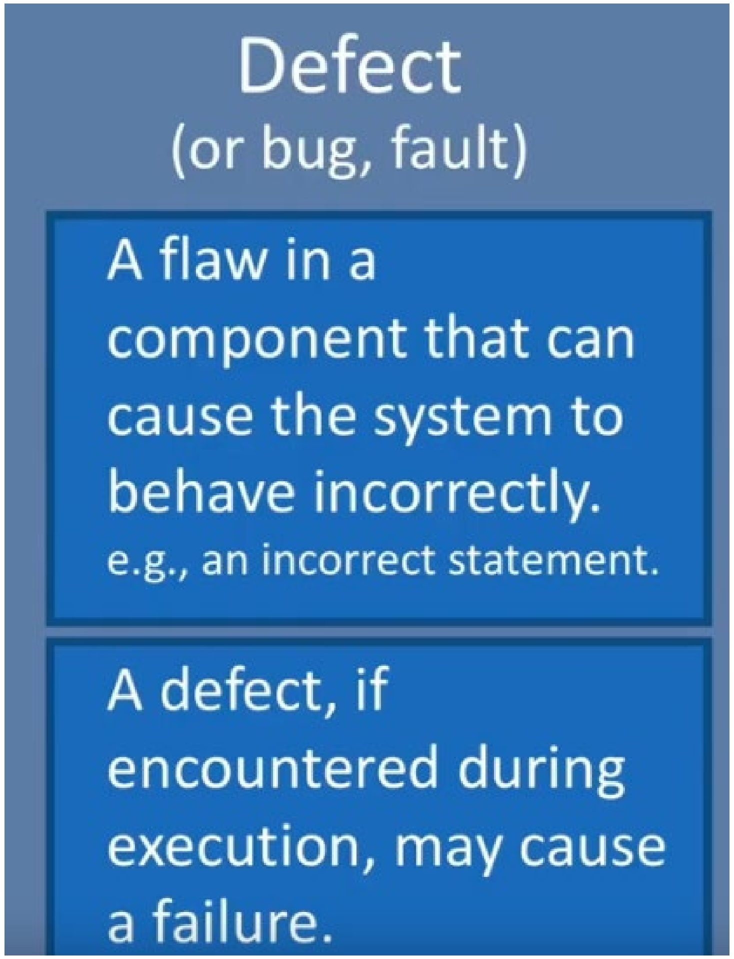
Miami-Dade's 4th-graders stand

→ Say Date today is 04/05/2021

Failure

→ This is news of 03/05/2021

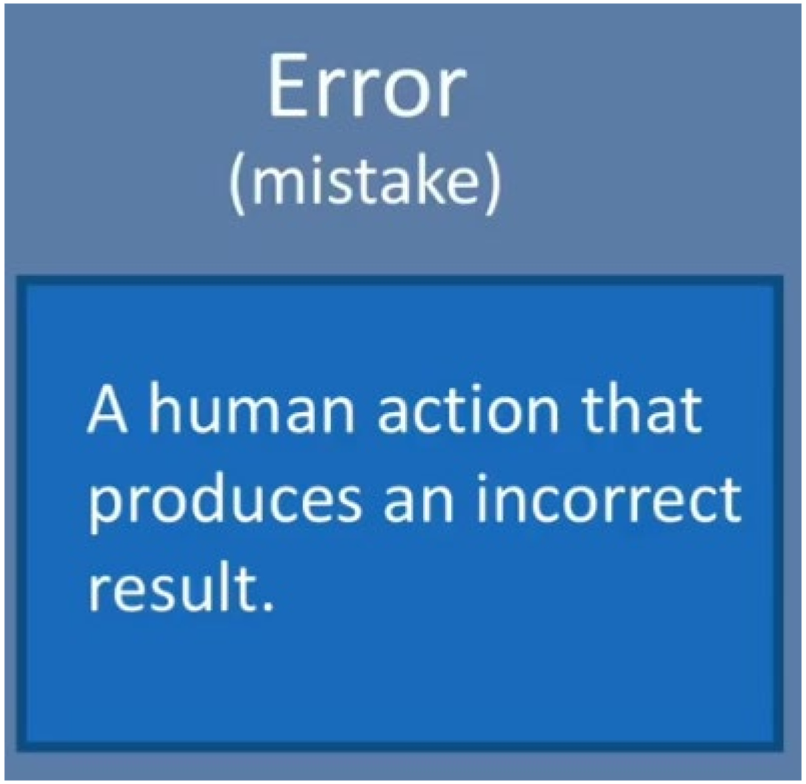
University of Victoria



Defect
(or bug, fault)

A flaw in a component that can cause the system to behave incorrectly.
e.g., an incorrect statement.

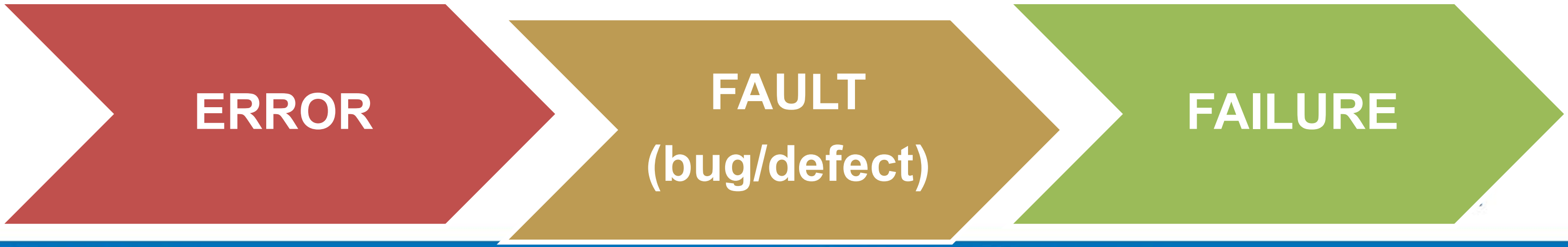
A defect, if encountered during execution, may cause a failure.



Error
(mistake)

A human action that produces an incorrect result.

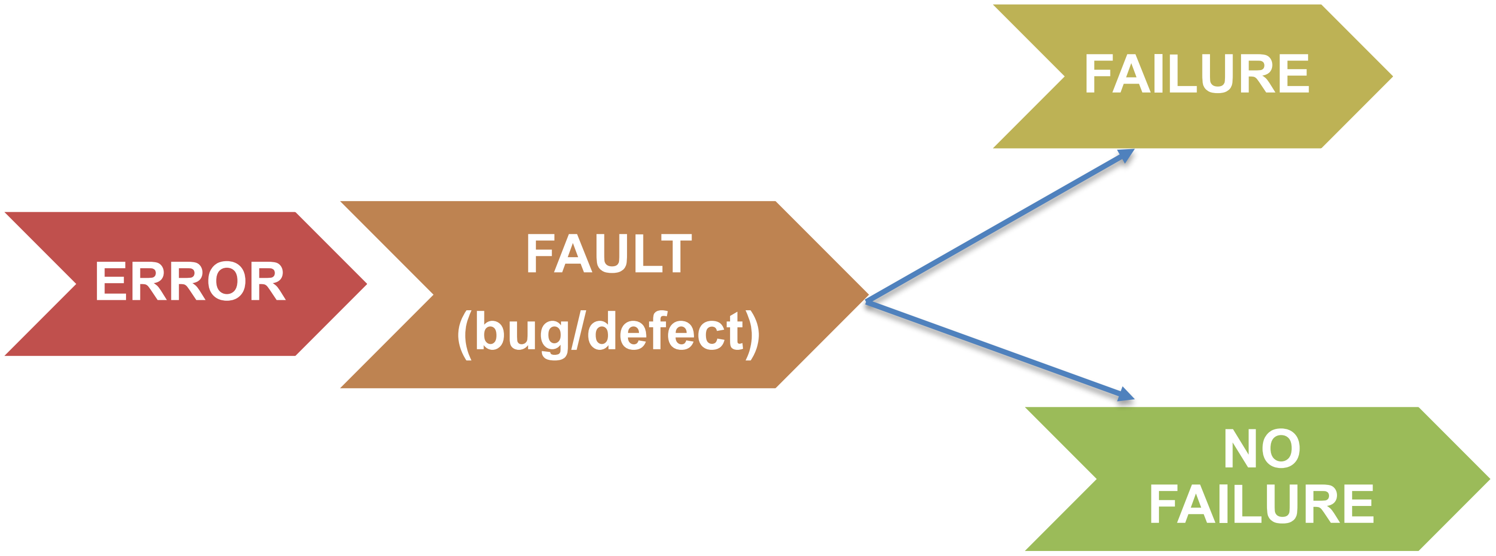
A mistake by a developer can lead to a fault in the source code that can eventually result in a failure.



Give Example Cases

1. Error: No, Fault: Yes, Failure: Yes
2. Error: No, Fault: Yes, Failure: No
3. Error: Yes, Fault: Yes, Failure: Yes
4. Error: Yes, Fault: Yes, Failure: No
5. Error: No, Fault: No, Failure: Yes

Depending on the test input, the fault may or may not manifest in a failure



Failures/faults/errors

- Failures are generally caused by faults.
- Faults are also called defects or bugs.
- A fault is the flaw in the component of the system that caused the system to behave incorrectly.
- A fault is technical and usually refers to source code, such as a comparison in an if statement that uses a "less than" operator (<) instead of a "greater than" operator (>). A broken connection is an example of a hardware fault.
- The existence of a fault in the source code does not necessarily lead to a failure. If the code containing the fault is never executed, it will never cause a failure.
- Error, also called mistake. An error is the human action (cause of a fault) that caused the system to run not as expected. Error is deviation from actual and expected value. It represents mistake made by people. E.g., Result is subtraction instead of addition, found during development.
- For example, a developer didn't cover an obscure condition because they misunderstood the requirement.
- Plugging a cable into the wrong socket is an example of a hardware mistake.

University of Victoria

Find Error, Fault, Failure

```
//Program to add two numbers
#include<stdio.h>
int main ()
{ int value1, value2, ans;
value1 = 5;
value2 = 3;
ans = value1 - value2;
printf("The addition is = %d.", ans);
return 0; }
```

When you compile and run this program you see the printed statement as below:

```
The addition is = 2.
```

Find Error, Fault, Failure

```
//Program to add two numbers
#include<stdio.h>
int main ()
{ int value1, value2, ans;
value1 = 5;
value2 = 3;
ans = value1 - value2;
printf("The addition is = %d.", ans);
return 0; }
```

Error: developer incorrectly interprets how addition is done or mistakenly inserts a minus symbol instead of plus.
Fault: minus symbol instead of plus
Failure: subtraction result instead of addition result.

When you compile and run this program you see the printed statement as below:

```
The addition is = 2.
```

University of Victoria

Program

```
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;
```

```
//Explanation
int[] exampleArray = {1,2,3,4,5};
int exampleArraySize = exampleArray.length;
System.out.print("This Java array size is: " + exampleArraySize );
//The Java array length example prints out the number 5
```

University of Victoria

Program

```
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;
```

Q) Identify the fault and fix it.

Q) Identify a test case where fault is executed but does not return a failure

Q) Identify a test case where fault is executed but does return a failure

Questions

Q) Identify fault and fix it.

Fault: for (int i = 1; i <x.length; i++)

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

Q) Identify a test case where fault is executed but does not return a failure.

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

Q) Identify a test case where fault is executed but does return a failure.

A) x = [0,7,2], fault executed, failure

FAULT/FAILURE/ERROR/BUG/DEFECT

VERIFICATION/VALIDATION

University of Victoria

Verification and Validation

- Verification:

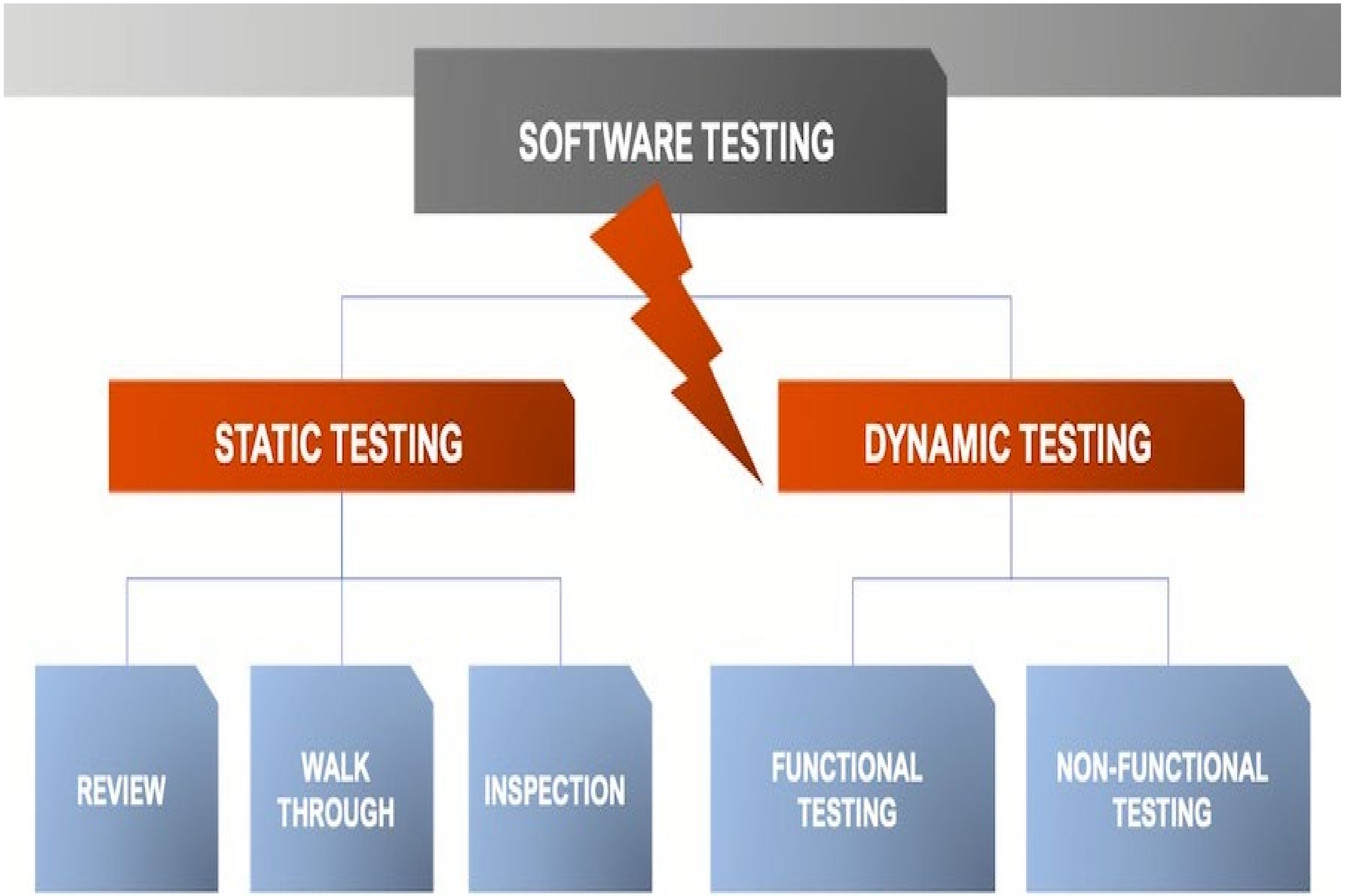
  “Are we building the product, right?”

- Validation:”

  “Are we building the right product?”

  (Is this what the user wants?)

University of Victoria


SOFTWARE TESTING
STATIC TESTING
DYNAMIC TESTING
REVIEW
WALK THROUGH
INSPECTION
FUNCTIONAL TESTING
NON-FUNCTIONAL TESTING

What is Static Testing?

- Static Testing is a type of software testing in which software application is tested without code execution.
- Manual or automated reviews of code, requirement documents and document design are done in order to find the errors.
- The main objective of static testing is to improve the quality of software applications by finding errors in early stages of software development process.

Examples of Work documents-

- Requirement specifications
- Design document
- Source Code
- Test Plans
- Test Cases
- Test Scripts
- Help or User document
- Web Page content

Testing Process

- Get the requirements
- Prepare Test Plan
- Create Test Scenarios
- Create Test cases for each Test Scenario
- Review meeting
- Change Test cases if required
- Upload the Test Cases
- Manual/Automation Testing

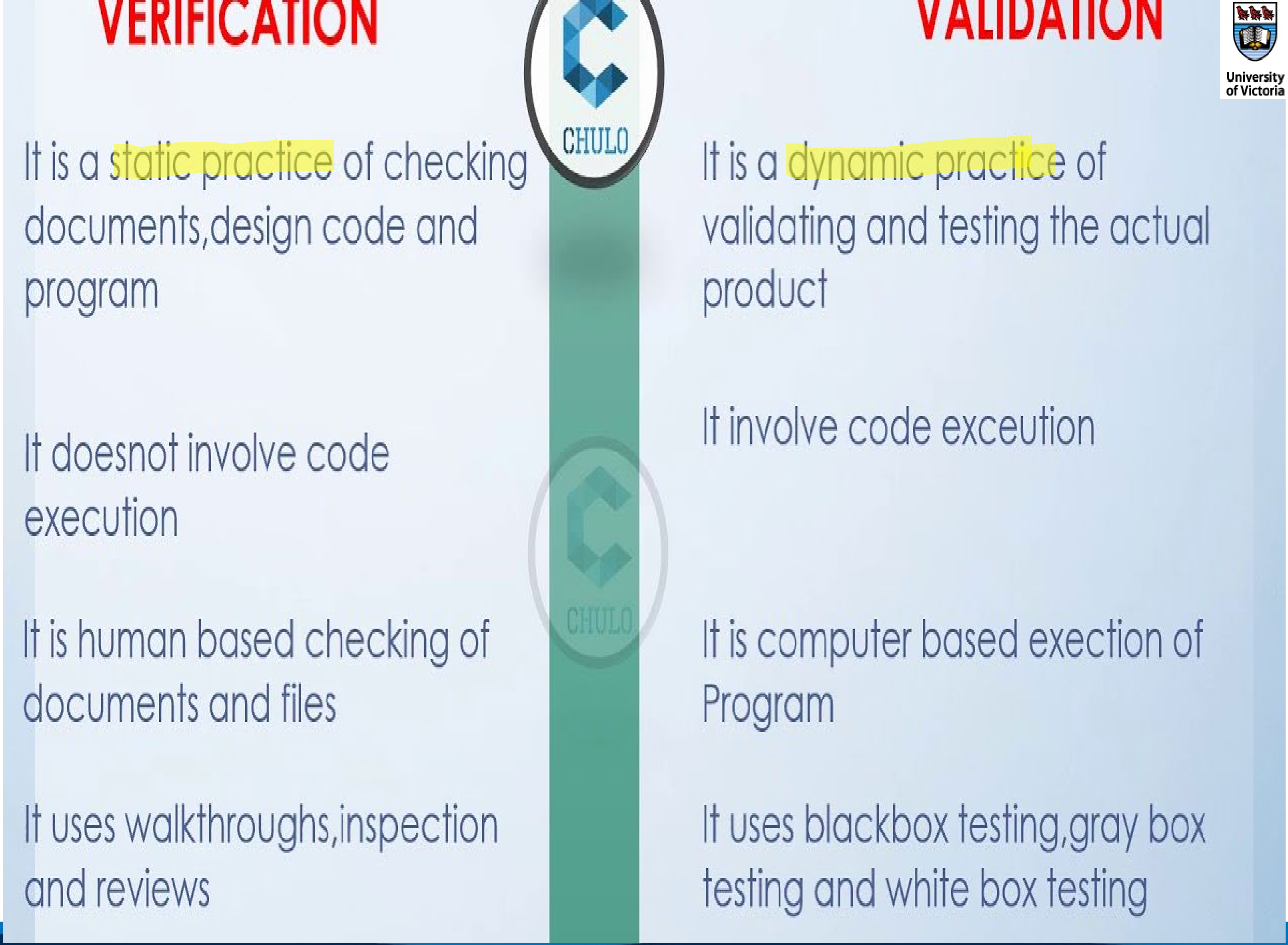
University of Victoria

What is Dynamic Testing?

- Under Dynamic Testing, a code is executed.
- It checks for functional behavior of software system, memory/cpu usage and overall performance of the system. Hence the name "Dynamic"

University of Victoria

University of Victoria



VERIFICATION	VALIDATION
It is a static practice of checking documents,design code and program	It is a dynamic practice of validating and testing the actual product
It doesnot involve code execution	It involve code exceution
It is human based checking of documents and files	It is computer based exection of Program
It uses walkthroughs,inspection and reviews	It uses blackbox testing,gray box testing and white box testing

Example of verification and validation

•In Software Engineering, consider the following specification

A clickable button with name Submet

•Verification would check the design doc and correcting the spelling mistake.

•Otherwise, the development team will create a button like

- So new specification is

A clickable button with name Submit

•Once the code is ready, Validation is done. A Validation test found –

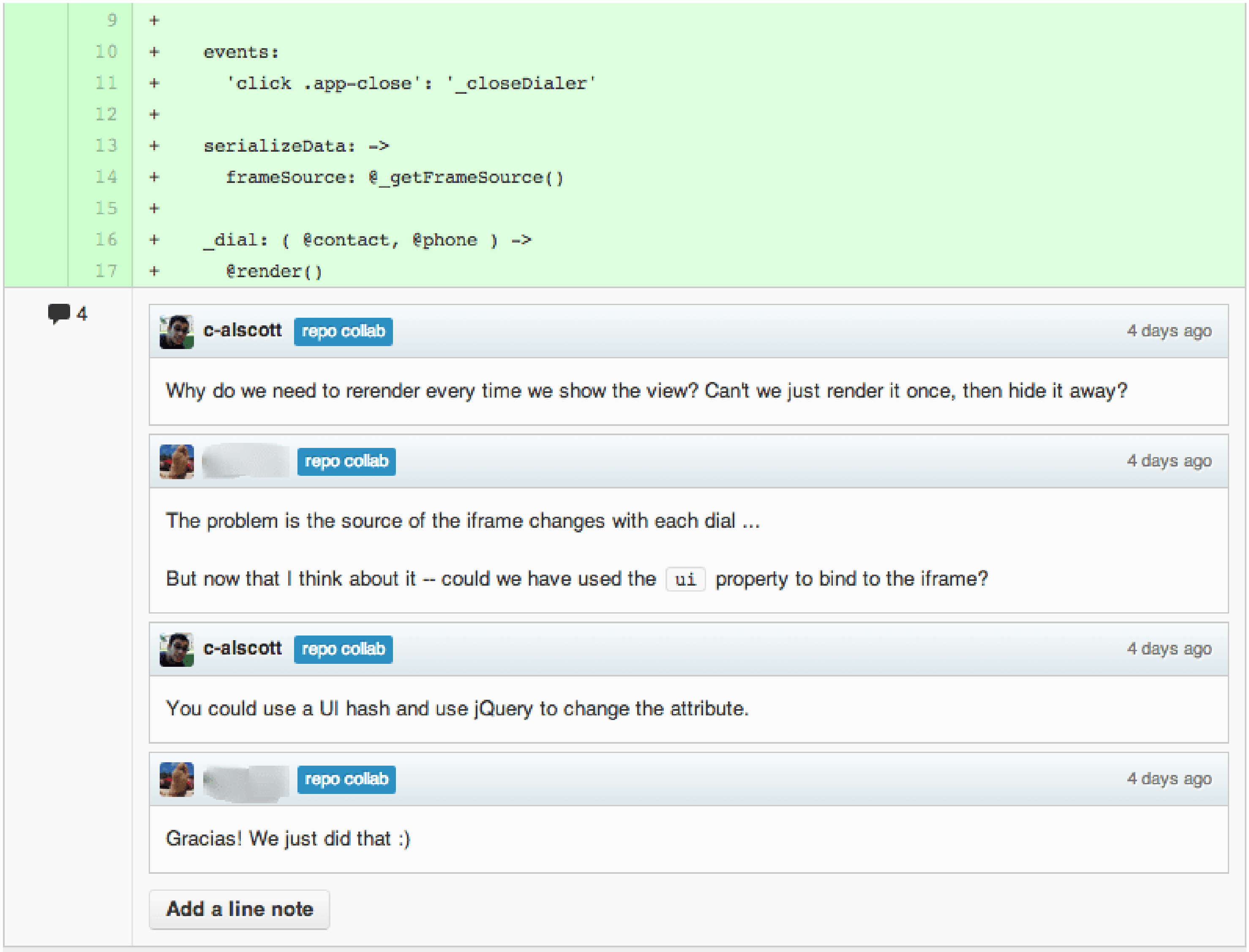
Button NOT Clickable

Owing to Validation testing, the development team will make the submit button clickable

University of Victoria

Verification Strategy	Explanation	Deliverable
Requirements Review	The study and discussions of the computer system requirements to ensure they meet stated user needs and are feasible.	Reviewed statement of requirements.
Design Review	The study and discussion of the computer system design to ensure it will support the system requirements.	System Design Document, Hardware Design Document.
Code Walkthrough	Informal analysis of the program source code to find defects and verify coding techniques.	Software ready for initial testing by the developer.
Code Inspection	Formal analysis of the program source code to find defects as defined by meeting system design specification.	Software ready for testing by the testing team.

Sample GitHub code review



```
+
+    events:
+      'click .app-close': '_closeDialer'
+
+    serializeData: ->
+      frameSource: @_getFrameSource()
+
+    _dial: ( @contact, @phone ) ->
+      @render()
```

4

c-alscott repo collab 4 days ago

Why do we need to rerender every time we show the view? Can't we just render it once, then hide it away?

repo collab 4 days ago

The problem is the source of the iframe changes with each dial ...

But now that I think about it -- could we have used the `ui` property to bind to the iframe?

c-alscott repo collab 4 days ago

You could use a UI hash and use jQuery to change the attribute.

repo collab 4 days ago

Gracias! We just did that :)

University of Victoria

Ref:https://dev.to/mporam/good-code-reviews-43kk

ERRORS IN REQUIREMENTS CAN BE FOUND USING VERIFICATION(REQUIREMENT TESTING)

Errors in ambiguous requirements

1) Weak wording:

“The system should not allow external users to access…”

does that mean that consultants working on-site get access or not?

The requirement is better phrased as

“The system will only allow employees to access…”

2) Pronoun ambiguity:

“When in a date field, the user should be able to open a pop-up calendar. *This* allows the user to…”

- What is the ***this*** that allows the user to do something? It could be the **date field** or the **calendar**.

University of Victoria

Adverbs ambiguity

- Try to be specific when describing the intended product characteristics so that all readers will share a common vision of what result they will have when they're done.
- Adverbs are **subjective** and result in ambiguity when **people interpret them differently.**
- **"The system must refresh the data reasonably quickly**"-Ambiguous
- **"The system must refresh the data within 0.5 seconds."-** Unambiguous
- "**Offer significantly better download speed."-**Ambiguous
- "**Offer download speed of 6 mbps"-**unambiguous

Ambiguity in Boundaries resolution

- **"Only supervisors may issue cash refunds greater than $50."**
- An analyst might derive several functional requirements from that business rule, such as the following:
  - 1. If the amount of the cash refund is less than $50, the system shall open the cash register drawer.
  - 2. If the amount of the cash refund is more than $50 and the user is a supervisor, the system shall open the cash register drawer. If the user is not a supervisor, the system shall display a message: "Call a supervisor for this transaction."
- But what if the amount of the cash refund is **exactly $50**? Is this a third, unspecified case? Or is it one of the two cases already described? If so, which one?

University of Victoria

You can resolve boundary ambiguities in one of three ways.

- **Tabular solution**
- **Less than or equal to solution**: The previous requirement #1 could be rewritten as,

"If the amount of the cash refund is less than or equal to $50, the system shall open the cash register drawer."

- **Inclusive, Exclusive solution**: Alternatively, you could use the words inclusive and exclusive to explicitly indicate whether the endpoints of a numerical range are considered to lie within the range or outside the range.

"If the amount of the cash refund is less $50, inclusive $50, the system shall open the cash register drawer."

University of Victoria