

# SOFTWARE ENGINEERING

C03001

## CHAPTER 9 — SOFTWARE QUALITY & QUALITY ASSURANCE

Anh Nguyen-Duc  
Quan Thanh Tho

# TOPICS COVERED

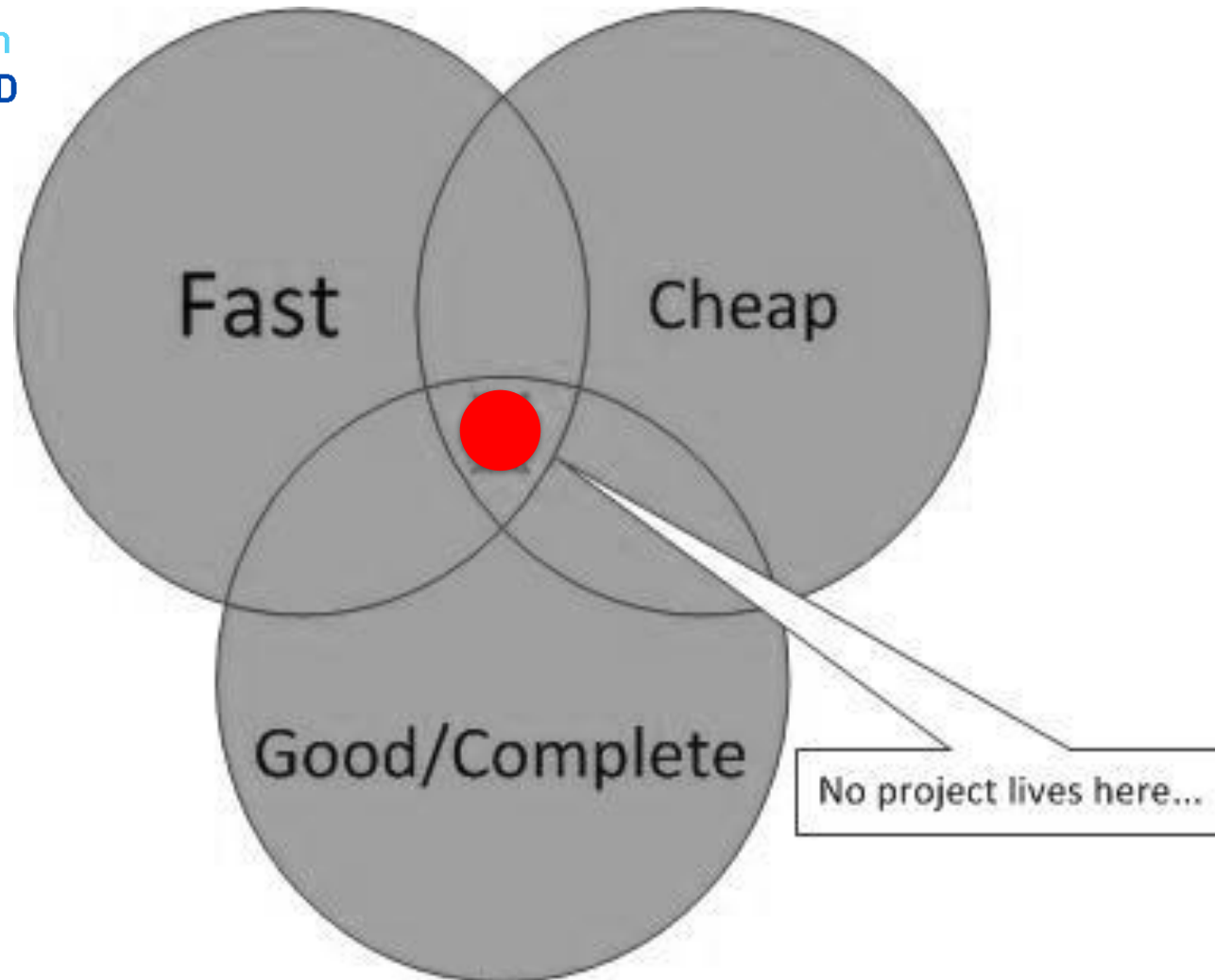
- ✓ Software Quality & its importance
- ✓ Software testing types and techniques
- ✓ Test plan and Test strategies



# THE IMPORTANCE OF SOFTWARE QUALITY

# WHY IS SOFTWARE QUALITY IMPORTANT?

**N** DIMECC Program  
NEED FOR SPEED



# DEFINITION OF QUALITY

- ✓ (ISO) defines **quality** as the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs (ISO8042:1994) or the degree to which a set of inherent characteristics fulfils requirements. (ISO9000:2000).
- ✓ **Conformance to requirements** means the project s processes and products meet written specifications.
- ✓ **Fitness for use** means a product can be used as it was intended.
- ✓ **Quality aspects:**
  - **product:** delivered to the customer
  - **process:** produces the software product
  - **resources:** (both the product and the process require resources)

# PROCESS QUALITY VS. PRODUCT QUALITY

- ✓ Quality can mean the difference between excellence and disaster
  - Airbus A400M Atlas crash in 2015, 4 killed



# PROCESS QUALITY VS. PRODUCT QUALITY

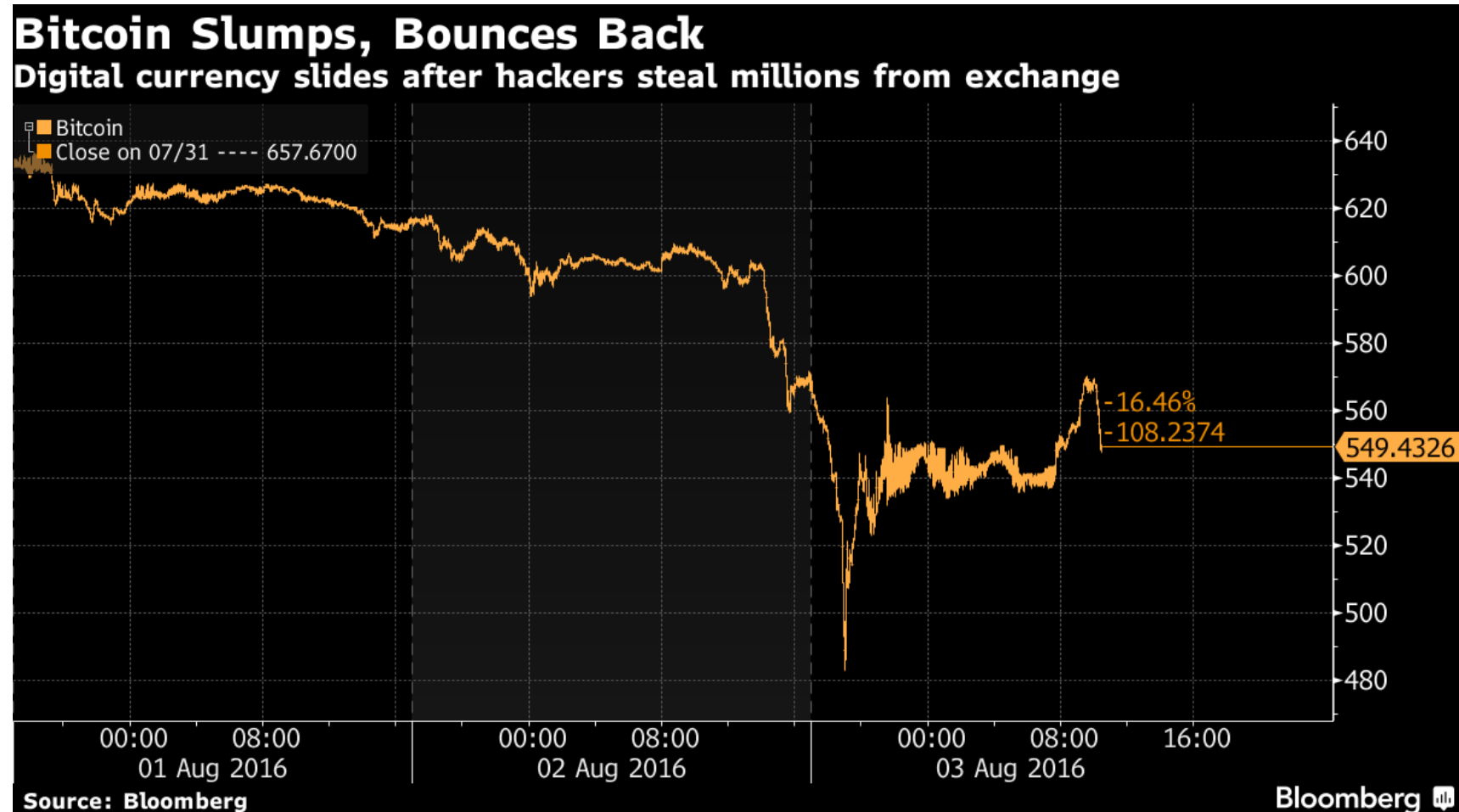
*“The black boxes attest to that there are no structural defects [with the aircraft], but we have a serious **quality** problem in the final assembly.”*



*“...either a weakness in the **test procedure** of planes before they fly, or a problem that results from the implementation of these procedures.”*

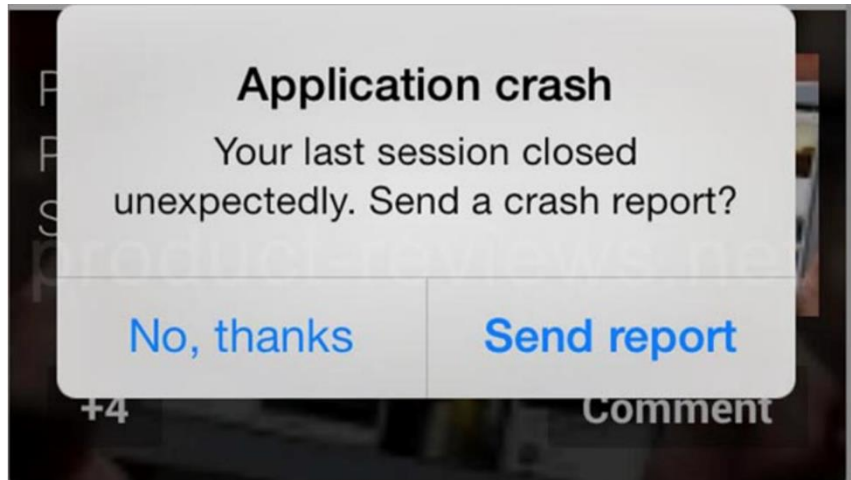
# PRODUCT OR PROCESS ISSUE?

- ✓ 8/2016: Security breach with Bitcoin cost 72 mil. Usd lost in market

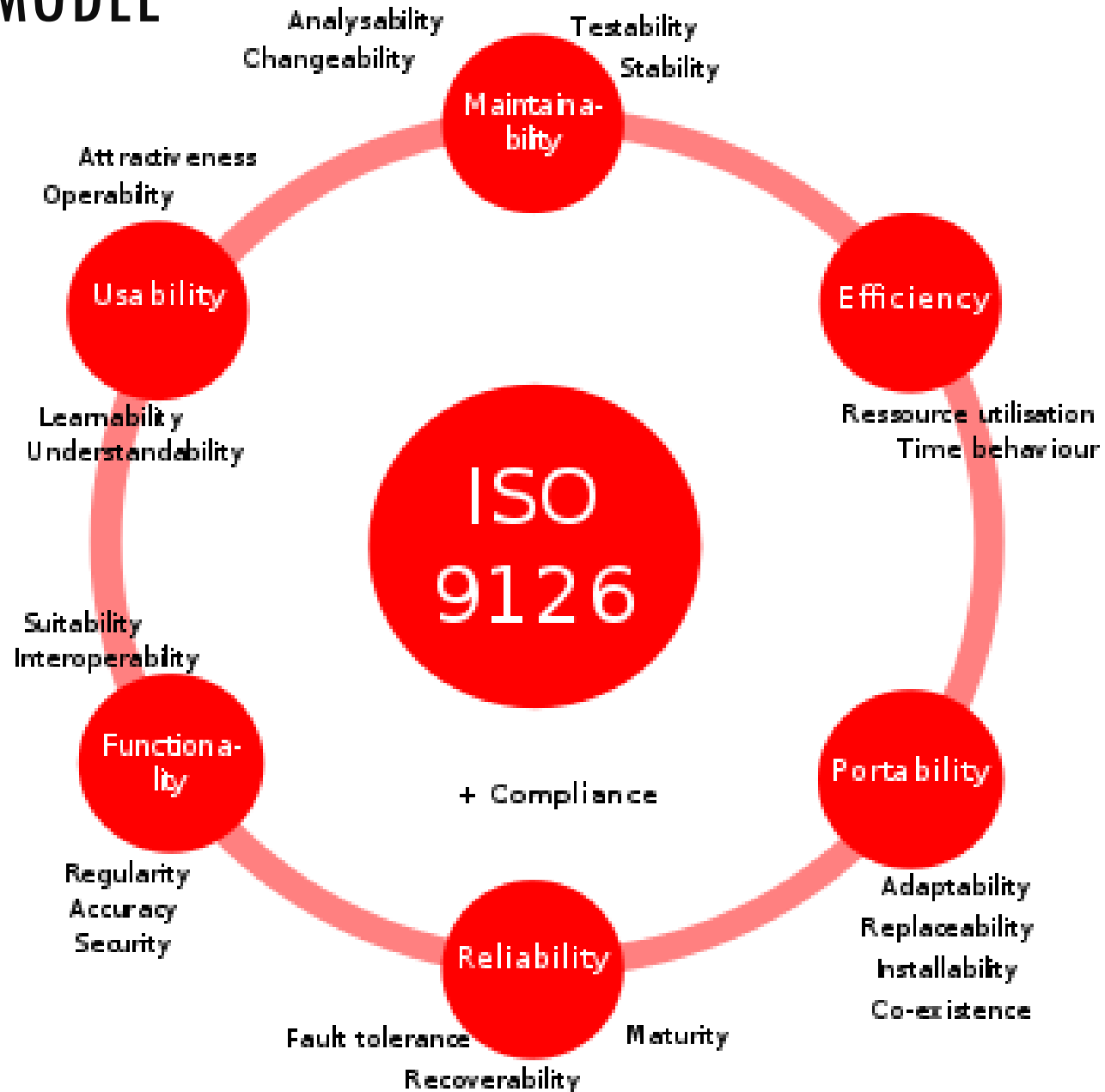




# SOFTWARE QUALITY ATTRIBUTES (1)



# SOFTWARE QUALITY MODEL





# SOFTWARE TESTING

# PROGRAM TESTING

- ✓ Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
- ✓ **Can reveal the presence of errors NOT their absence.**
- ✓ Testing is part of a more general verification and validation process, which also includes static validation techniques.

# PROGRAM TESTING GOALS

- ✓ To demonstrate to the developer and the customer that the software meets its requirements.
  - validation testing
  
- ✓ To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification.
  - defect testing

The Product  
of Testing  
is  
**CONFIDENCE**

# PSYCHOLOGY OF TESTING (1)

- ✓ A program is its programmer's baby!
  - Trying to find errors in one's own program is like trying to find defects in one's own baby.
  - It is best to have someone other than the programmer doing the testing.
- ✓ Tester must be highly skilled, experienced professional.
- ✓ It helps if he or she possesses a diabolical mind.

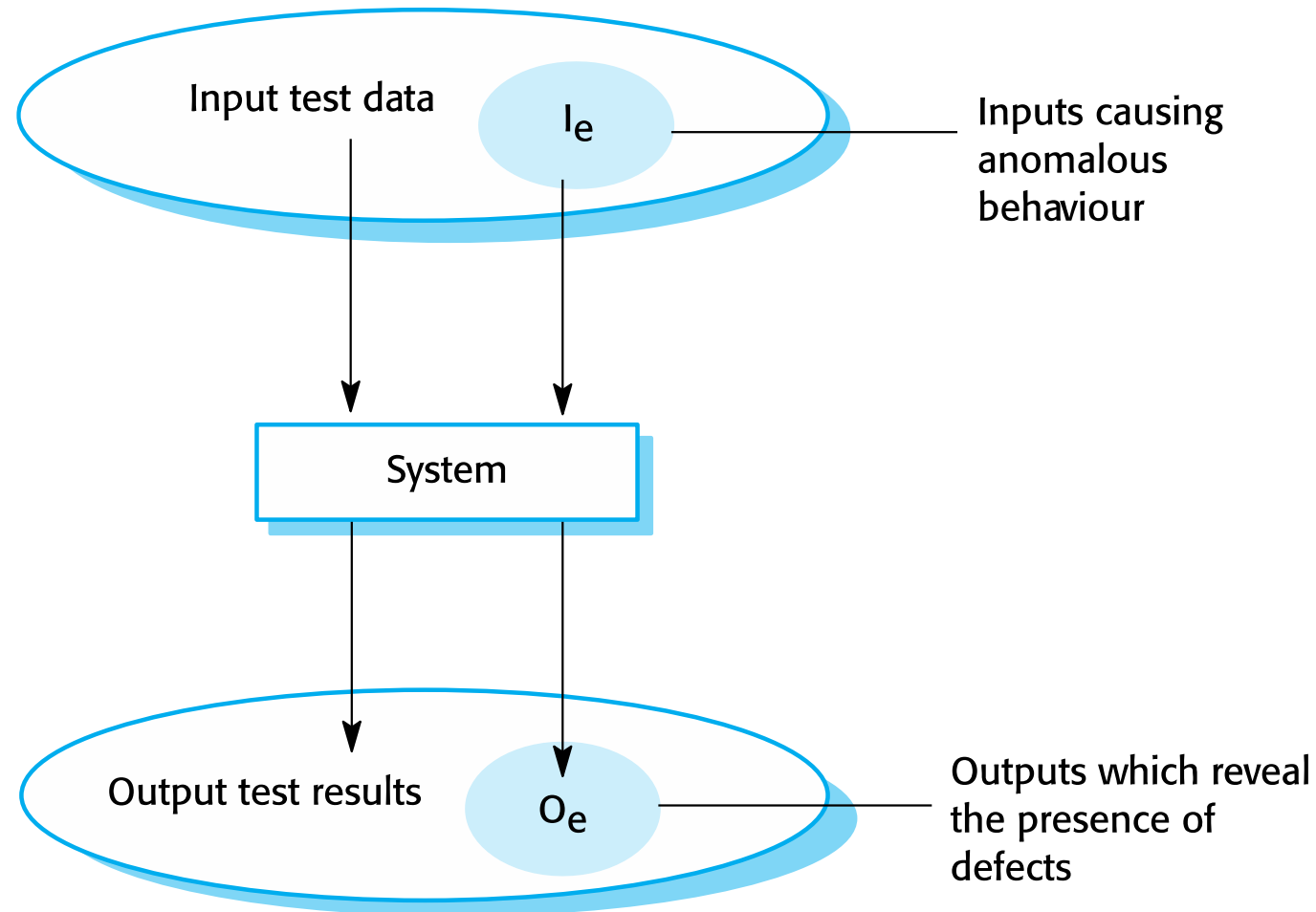
## PSYCHOLOGY OF TESTING (2)

- ✓ Testing achievements depend a lot on what are the goals.
- ✓ Myers says (79):
  - If your goal is to **show absence of errors**, you will not discover many.
  - If you are trying to **show the program correct**, your subconscious will manufacture safe test cases.
  - If your goal is to **show presence of errors**, you will discover large percentage of them.

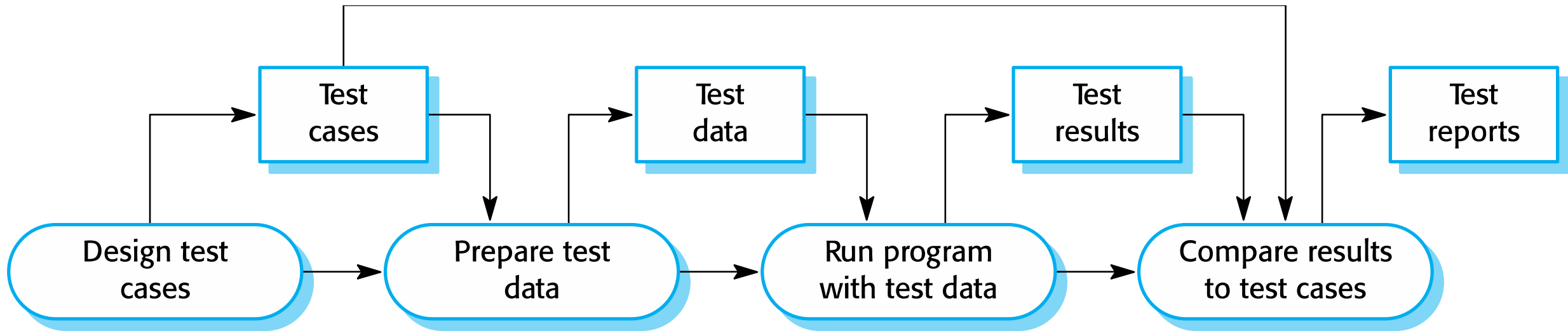
Testing is the process of executing a program with the intention of finding errors (G. Myers)



# AN INPUT-OUTPUT MODEL OF PROGRAM TESTING



# A MODEL OF THE SOFTWARE TESTING PROCESS



# STAGES OF TESTING

- ✓ Development testing
  - the system is tested during development to discover bugs and defects.
- ✓ Release testing
  - a separate testing team test a complete version of the system before it is released to users.
- ✓ User testing
  - users or potential users of a system test the system in their own environment.

# DEVELOPMENT TESTING

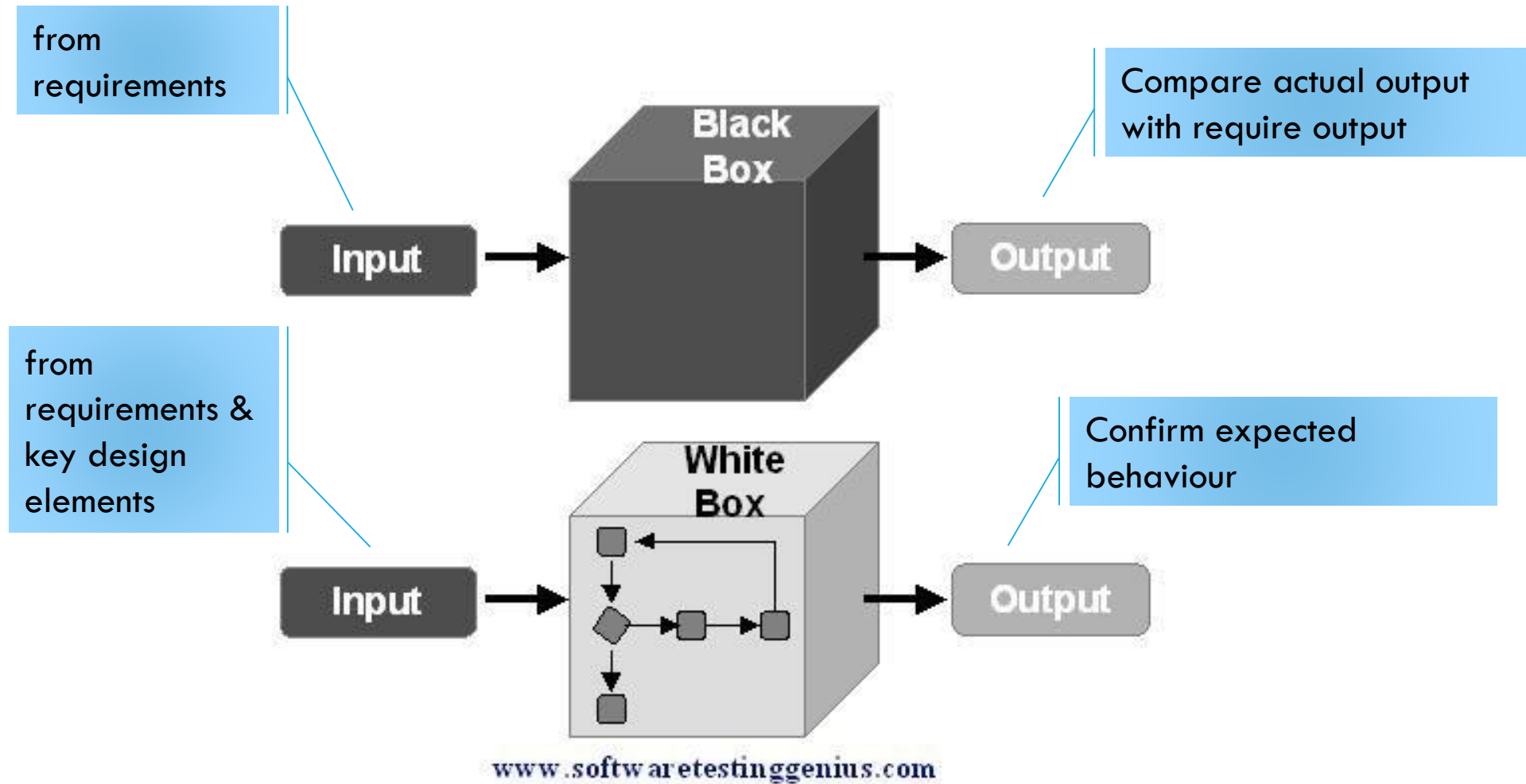
*carried out by the team developing the system.*

- ✓ Unit testing:
  - for individual program units or object classes
  - focus on testing the functionality of objects or methods.
- ✓ Component testing:
  - several individual units are integrated to create composite components
  - focus on testing component interfaces.
- ✓ System testing:
  - some or all of the components in a system are integrated and the system is tested as a whole
  - focus on testing component interactions.

# UNIT TESTING

- ✓ Unit testing is the process of testing individual components in isolation.
- ✓ It is a defect testing process.
- ✓ Units may be:
  - Individual functions or methods within an object
  - Object classes with several attributes and methods
  - Composite components with defined interfaces used to access their functionality.

# UNIT TESTING: BLACK-/WHITE-BOX TEST



***Gray-box: mix of black- and white-box testing***

```
for (i=0; i<numrows; i++)  
    for (j=0; j<numcols; j++);  
    pixels++;
```

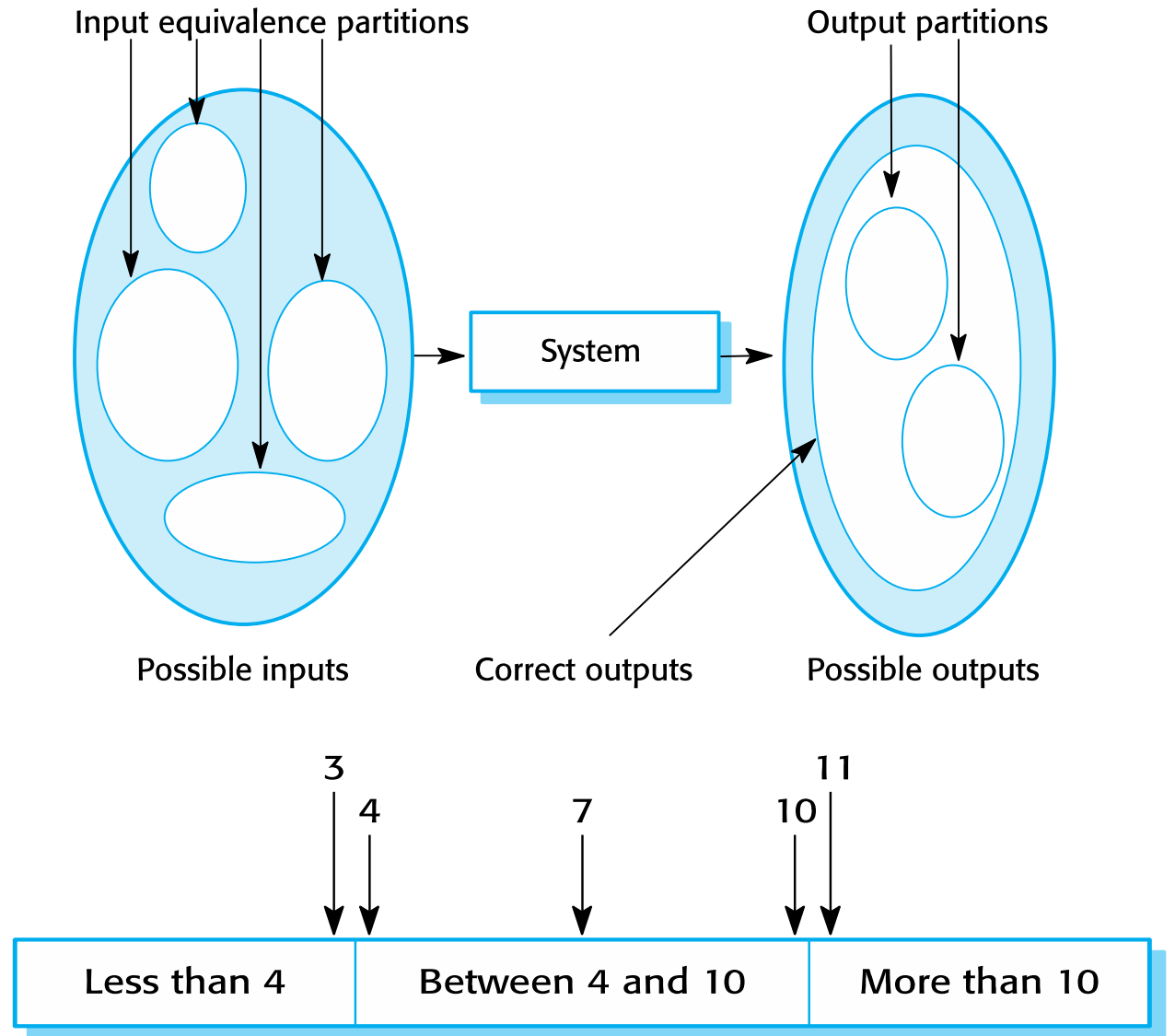
```
int minval(int *A, int n) {  
    int currmin;  
  
    for (int i=0; i<n; i++)  
        if (A[i] < currmin)  
            currmin = A[i];  
    return currmin;  
}
```

```
switch (i) {  
    case 1:  
        do_something(1); break;  
    case 2:  
        do_something(2); break;  
    case 3:  
        do_something(1); break;  
    case 4:  
        do_something(4); break;  
    default:  
        break;  
}
```



# EQUIVALENCE PARTITIONING & BOUNDARY VALUE ANALYSIS

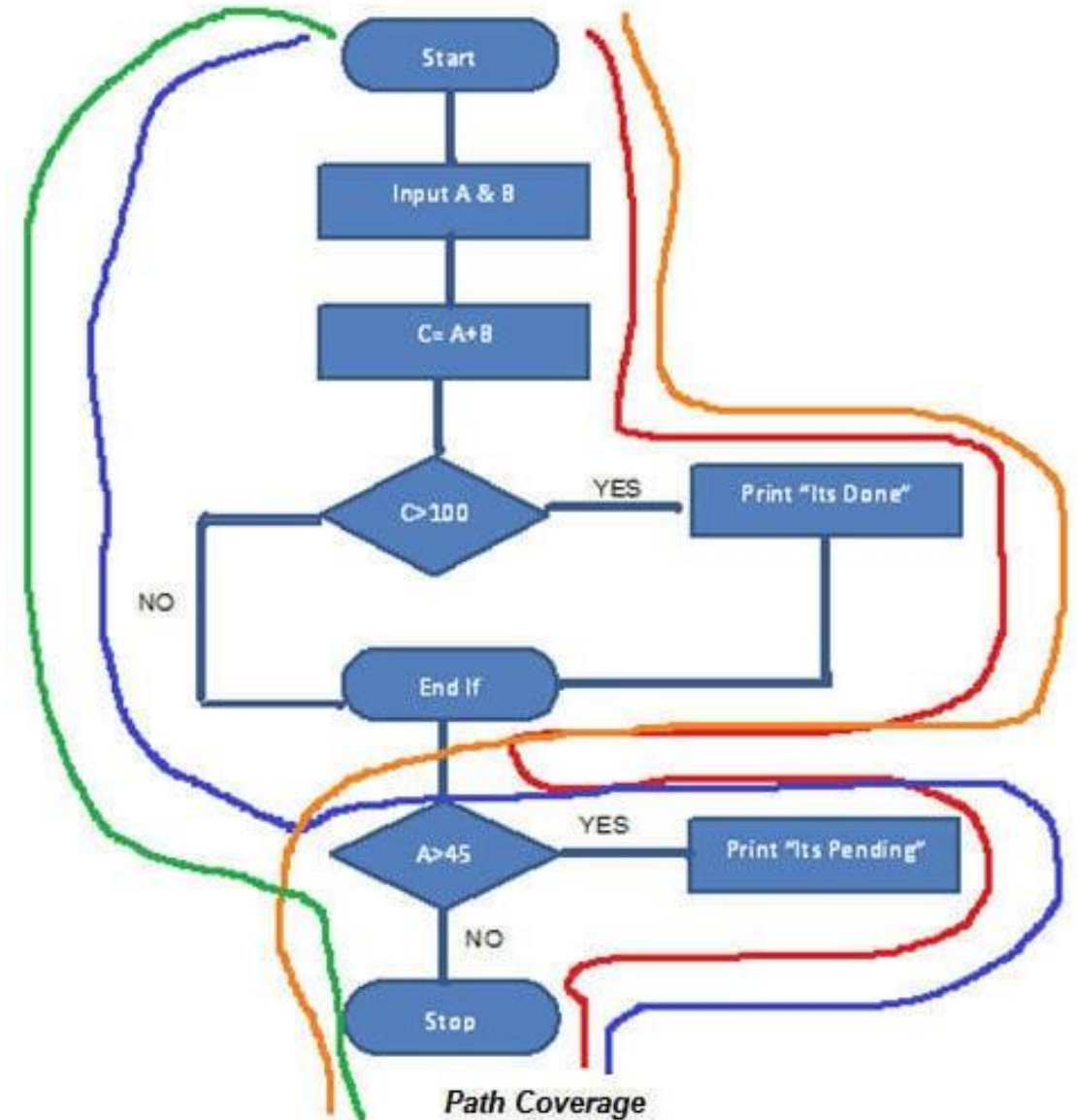
```
public int multi(int x,int y){  
    int z;  
    z=x*y;  
    return z;  
}
```



Number of input values

# WHITE-BOX TESTING

- ✓ Statement coverage
- ✓ Branch coverage
- ✓ Path coverage



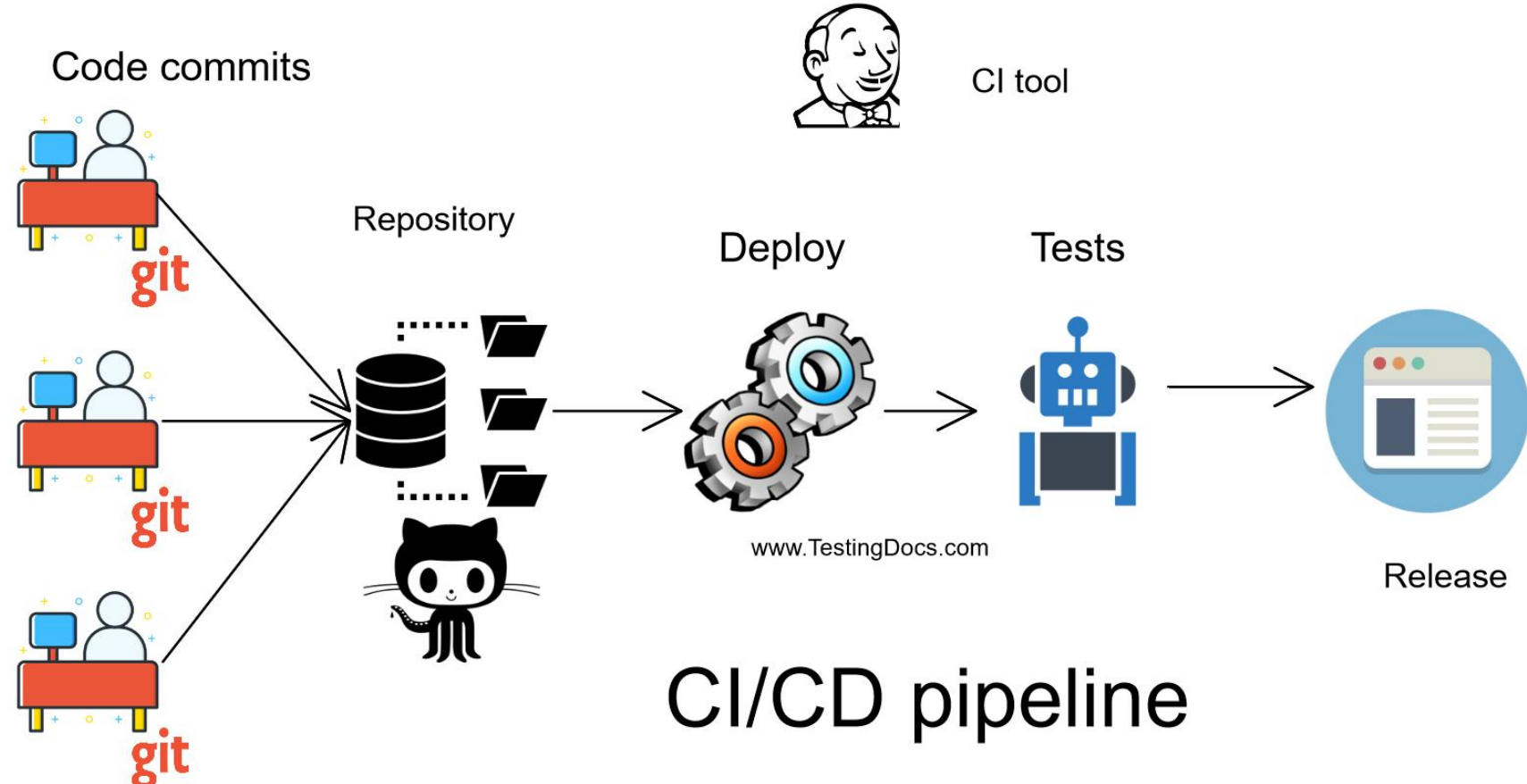
# REGRESSION TESTING

*Test the system to check that changes have not 'broken' previously working code.*

- ✓ Better with automated testing
- ✓ All tests are re-run every time a change is made to the program.
- ✓ Tests must run 'successfully' before the change is committed.

# AUTOMATED TESTING

- ✓ Whenever possible, unit testing should be automated
- ✓ Use of a test automation framework (such as JUnit)



# SYSTEM TESTING

*System testing during development = to create a version of the system and then testing the integrated system.*

- ✓ Focus on testing the interactions between components.
  - System testing checks that components are compatible, interact correctly and transfer the right data at the right time across their interfaces.
  
- ✓ And tests the emergent behaviour of a system.



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Tests are designed to verify each of the functionalities



Tests are designed to verify that all the modules function individually as desired within the systems



The idea here is to ensure that individual modules function correctly within the whole system.



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

## Tests are designed to verify

- **look-and-feel** the interface to the users of an application system
- different components such as icons, menu bars, dialog boxes, scroll bars, list boxes, and radio buttons

The GUI can be utilized to test the functionality behind the interface, such as:

- accurate response to database queries
- usefulness of the on-line help, error messages, tutorials, and user manuals

The usability characteristics of the GUI is tested, which includes the following

- **Accessibility:** Can users enter, navigate, and exit with relative ease?
- **Responsiveness:** Can users do what they want and when they want in a way that is clear?
- **Efficiency:** Can users do what they want to with minimum number of steps and time?
- **Comprehensibility:** Do users understand the product structure with a minimum amount of effort?



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

## Accessibility Fundamentals

### Introduction to Accessibility

#### Video Introduction

#### Accessibility is About People

#### Components of Web Accessibility

#### Accessibility Principles

#### Digital Accessibility Courses

### Languages/Translations

- English (Original)
- العربية
- Čeština
- Español
- Français
- Bahasa Indonesia
- 한국어
- Русский Язык
- 简体中文

#### All Translations

#### Translating WAI Resources

## Introduction to Web Accessibility

### Summary

When websites and web tools are properly designed and coded, people with disabilities can use them. However, currently many sites and tools are developed with accessibility barriers that make them difficult or impossible for some people to use.

Making the web accessible benefits individuals, businesses, and society. International web standards define what is needed for accessibility.

### Page Contents

- [Accessibility in Context](#)
- [What is Web Accessibility](#)
- [Accessibility is Important for Individuals, Businesses, Society](#)
- [Making the Web Accessible](#)
- [Evaluating Accessibility](#)
- [Examples](#)
- [For More Information](#)

### Related Resource





Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

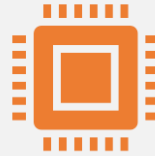
Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing



Tests are designed to verify the ability of the system to inter-operate with third party products.



Compatibility tests are tests that verify that the system works the same way across different platforms, operating systems, data base management systems



Backward compatibility tests verify that the current software build flawlessly works with older version of platforms

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Robustness means how much sensitive a system is to erroneous input and changes its operational environment

Tests in this category are designed to verify how gracefully the system behaves in error situations and in a changed operational environment

Some types of robustness tests:

- The tests include providing invalid input data to the system and observing how the system reacts to the invalid input.
- **Power cycling** tests are executed to ensure that, when there is a power glitch in a deployment environment, the system can recover from the glitch to be back in normal operation after power is restored.
- **Availability tests** are designed to is to verify that the system gracefully and quickly recovers from hardware and software failures without adversely impacting the operation of the system.

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Tests are designed to determine the performance of the actual system compared to the expected one

Examples of Performance metrics: response time, execution time, throughput, resource utilization and traffic rate

For Example: If the objective is to evaluate the response time, then one needs to capture

- End-to-end response time (as seen by external user)
- CPU time
- Network connection time
- Database access time
- Network connection time
- Waiting time

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

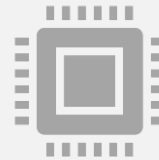
Safety Assurance

Regression Tests

Acceptance Testing



Tests are designed to verify that the system can scale up to its engineering limits while continuing to meet performance requirements



Scaling tests are conducted to ensure that the system response time remains the same, or increases by a small amount, as the number of users are increased.



There are three major causes of these limitations:

- data storage limitations
- network bandwidth limitations
- speed limit

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

The system is deliberately stressed by pushing it to and beyond its specified limits

It ensures that the system can perform acceptably under **worst-case conditions**, under an expected peak load.

Targeted to bring out the problems associated with: Memory leak and Buffer allocation and memory carving

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Load and stability testing typically involves exercising the system with virtual users and measuring the performance to verify whether the system can support the anticipated load



When a large number of users are introduced a number of problems are likely to occur:

the system slows  
down

the system encounters  
functionality problems

the system crashes  
altogether

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

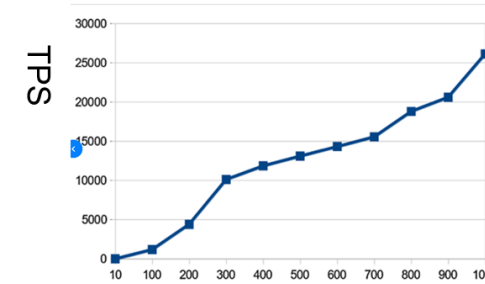
Acceptance Testing

Scalability

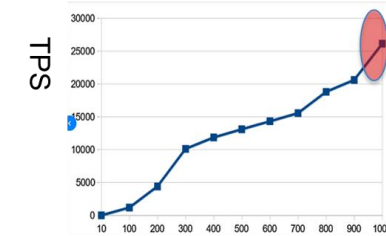
Stress

Load and  
stability

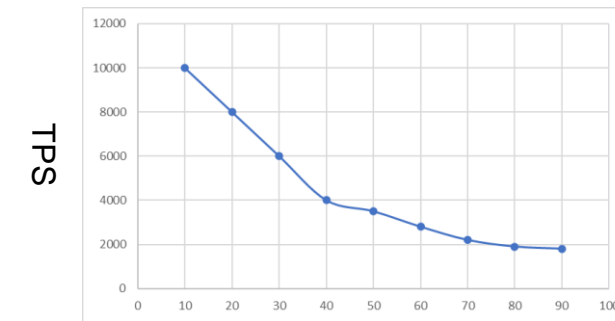
TPS(Transaction per second)



Servers



Servers



Time



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests



Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Testing of security requirements that concerns confidentiality, integrity, availability, authentication, authorization, nonrepudiation, and;

The testing to validate the ability of the software to withstand attack (resiliency).



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

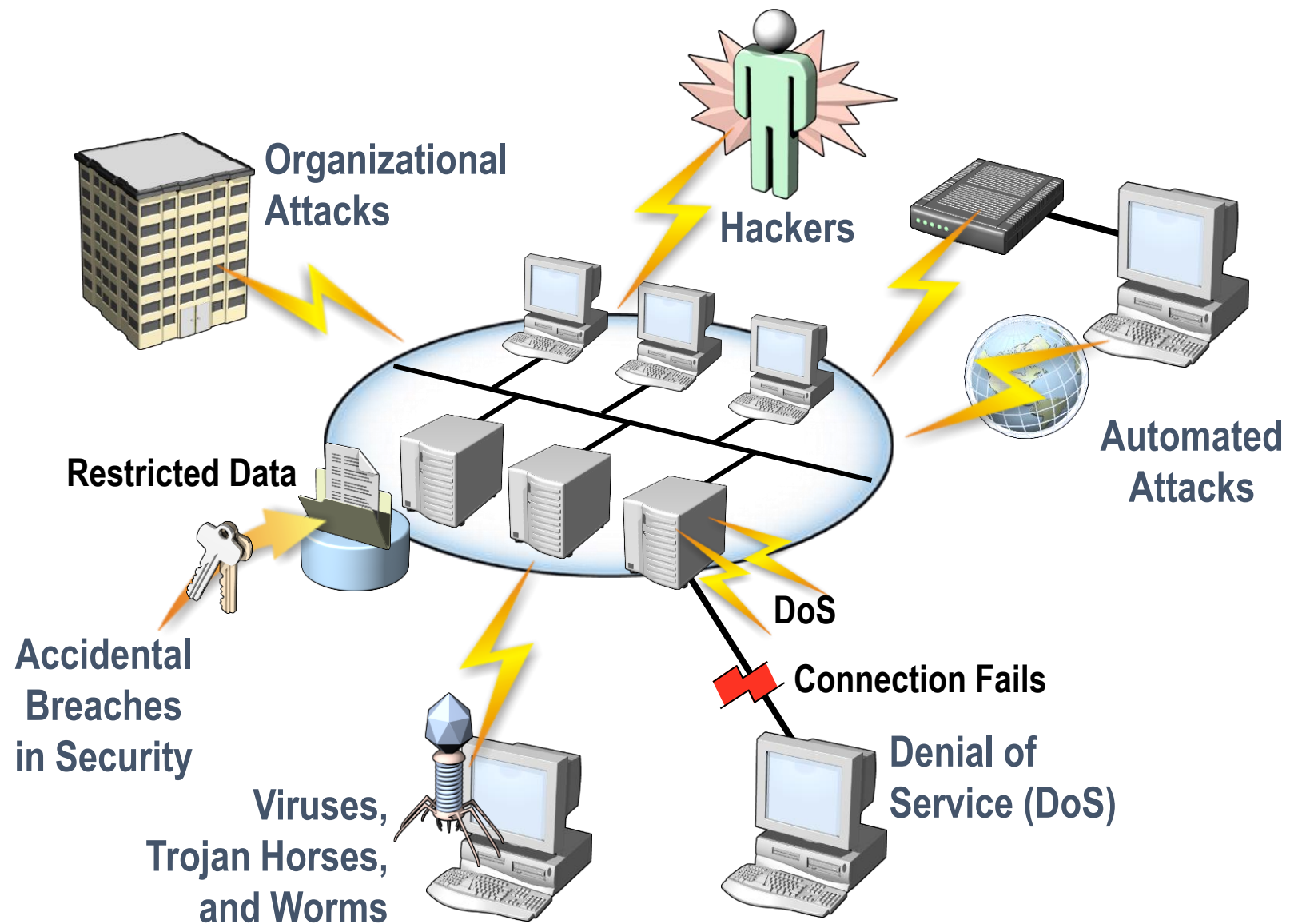
Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Network

Host

Application

Threats against  
the network

*Spoofed packets, etc.*

Threats against the host

*Buffer overflows, illicit paths, etc.*

Threats against the application

*SQL injection, XSS, input tampering, etc.*

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

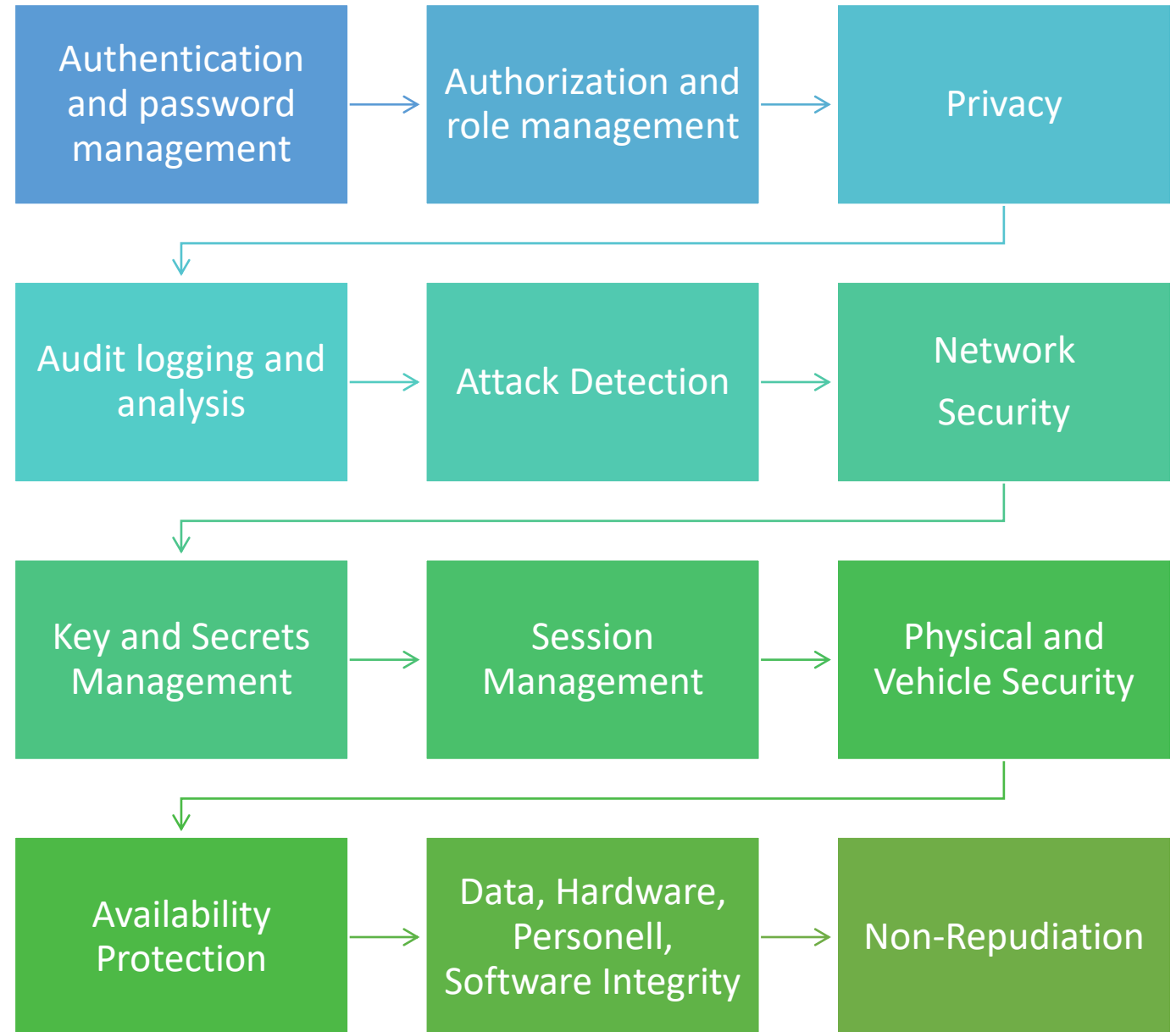
Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing



# Privacy and GDPR

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

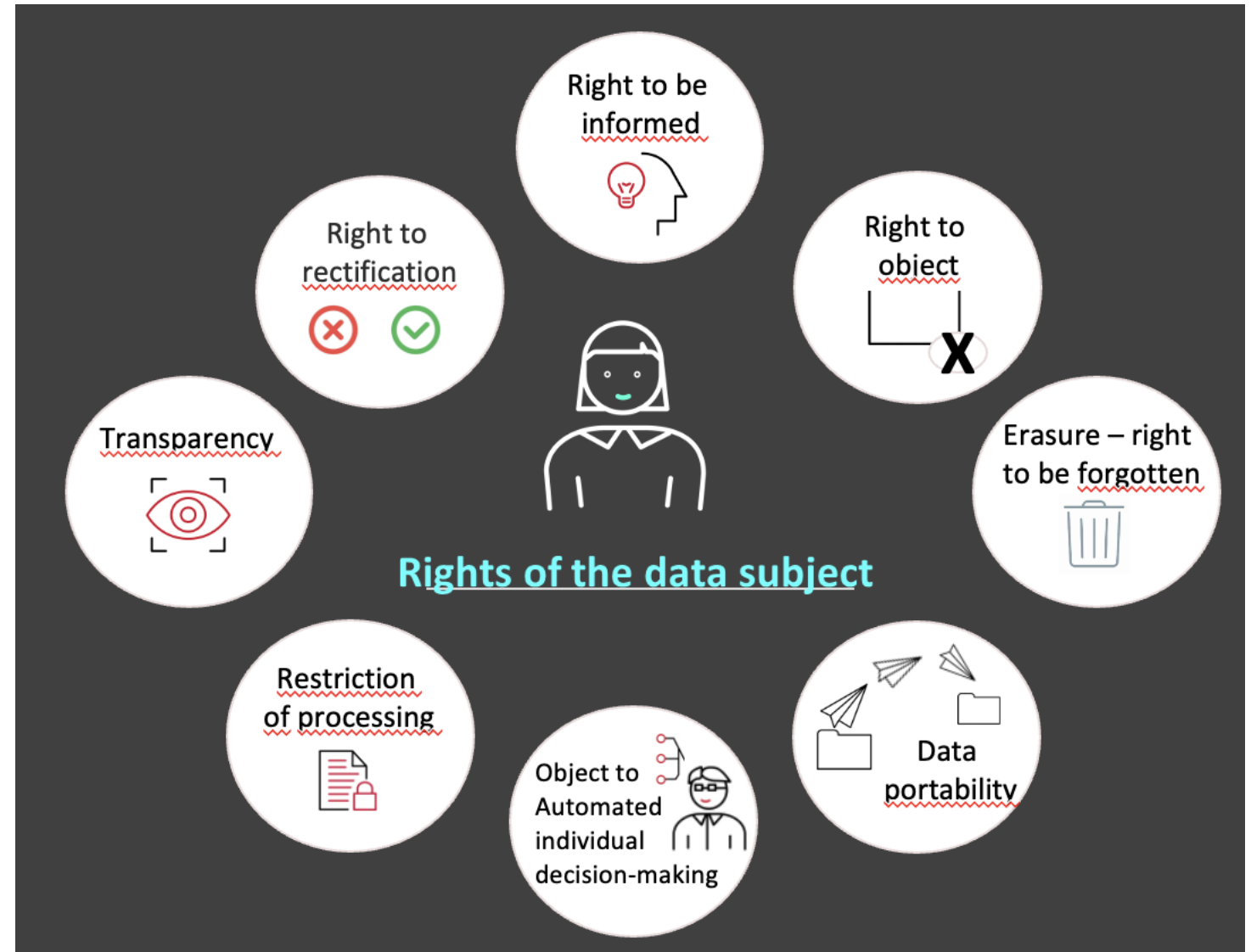
Security Testing

Regulatory Tests

Safety Assurance

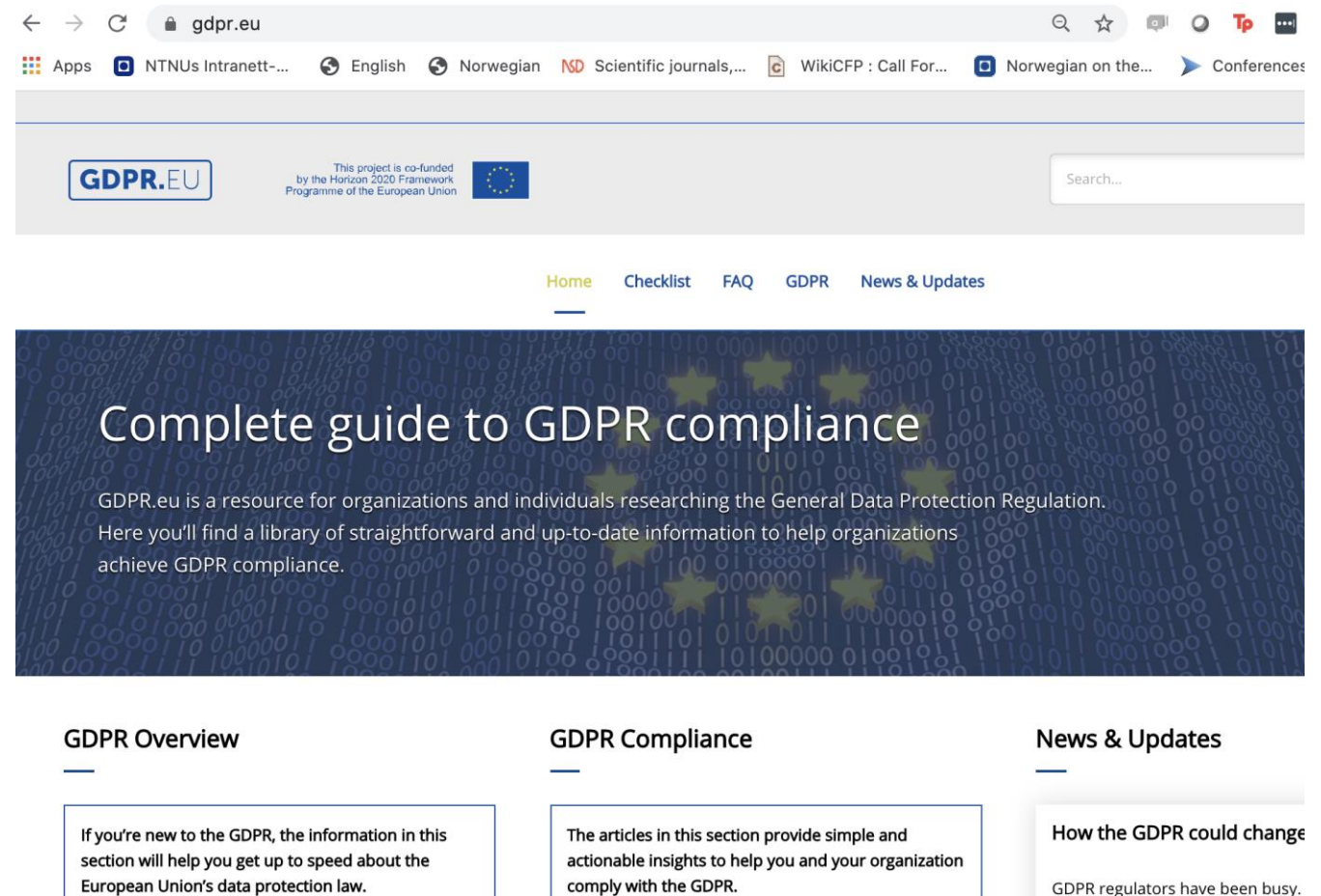
Regression Tests

Acceptance Testing

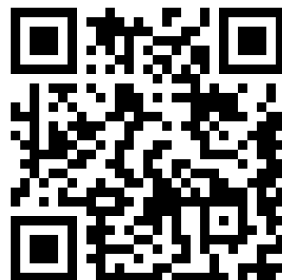




# Privacy and GDPR



The screenshot shows the GDPR.EU website homepage. The browser address bar displays 'gdpr.eu'. The page features a navigation menu with links: Home, Checklist, FAQ, GDPR, and News & Updates. A large banner titled 'Complete guide to GDPR compliance' states: 'GDPR.eu is a resource for organizations and individuals researching the General Data Protection Regulation. Here you'll find a library of straightforward and up-to-date information to help organizations achieve GDPR compliance.' Below the banner, three sections are visible: 'GDPR Overview' (describing the site's purpose), 'GDPR Compliance' (offering actionable insights), and 'News & Updates' (mentioning that regulators have been busy).



<https://gdpr.eu/tag/gdpr/>

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

The final system is shipped to the regulatory bodies in those countries where the product is expected to be marketed



The regulatory agencies are usually interested in identifying flaws in software that have potential safety consequences



The safety requirements are primarily based on their own published standards

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

Software *safety* is defined in terms of hazards

A *hazard* is a state of a system or a physical situation which when combined with certain environmental conditions, could lead to an *accident* that results in death, injury, illness, damage or loss of property, or harm to the environment.

A software in isolation cannot do physical damage. However, a software in the context of a system and an embedding environment could be vulnerable

There are two basic tasks performed by a **safety assurance** engineering team:

Provide methods for identifying, tracking, evaluating, and eliminating hazards associated with a system

Ensure that safety is embedded into the design and implementation in a timely and cost effective manner, such that the risk created by the user/operator error is minimized

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

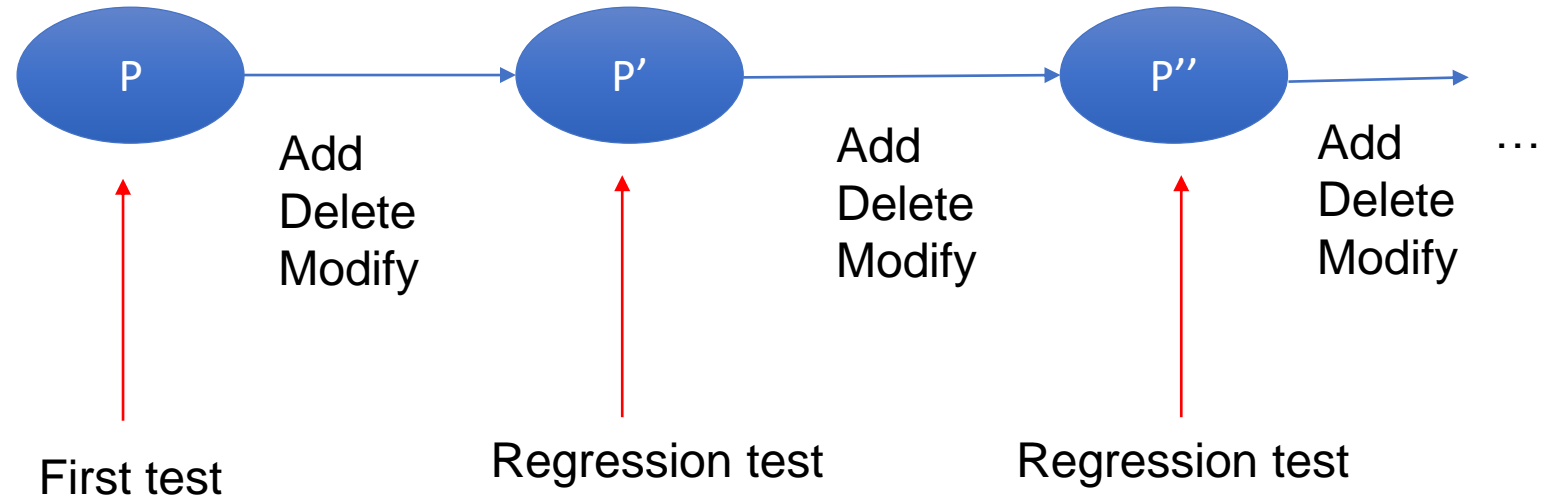
Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing



One of the main goals is to verify that no defect has been introduced into the unchanged portion of a system due to changes made elsewhere in the system with maximized likelihood of uncovering new defects and reduced cost of testing



Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing



### Test revalidation

- Unchanged code continues to behave correctly

### (Regression) test selection

- Verify changed and impacted code (corrective)
- Verify new structure and requirements (progressive)

### Test minimization

- Remove redundant test cases
- Are test cases obsolete?

### Test prioritization

- Rank test cases and run them according to available resources
- Do not remove test cases from the set

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

Safety Assurance

Regression Tests

Acceptance Testing

## Retest all

- Can be costly, unless test execution is highly automated

## Random selection

- Better than no test
- Hard to ensure coverage of the changed code

## Selecting modification traversing tests

- A technique that **does not discard any test** that will **traverse a modified or impacted statement** is known as “**safe**” regression test selection technique

Functionality Tests

GUI Testing

Interoperability Tests

Robustness Tests

Performance Tests

Scalability Tests

Stress Tests

Load and Stability Tests

Security Testing

Regulatory Tests

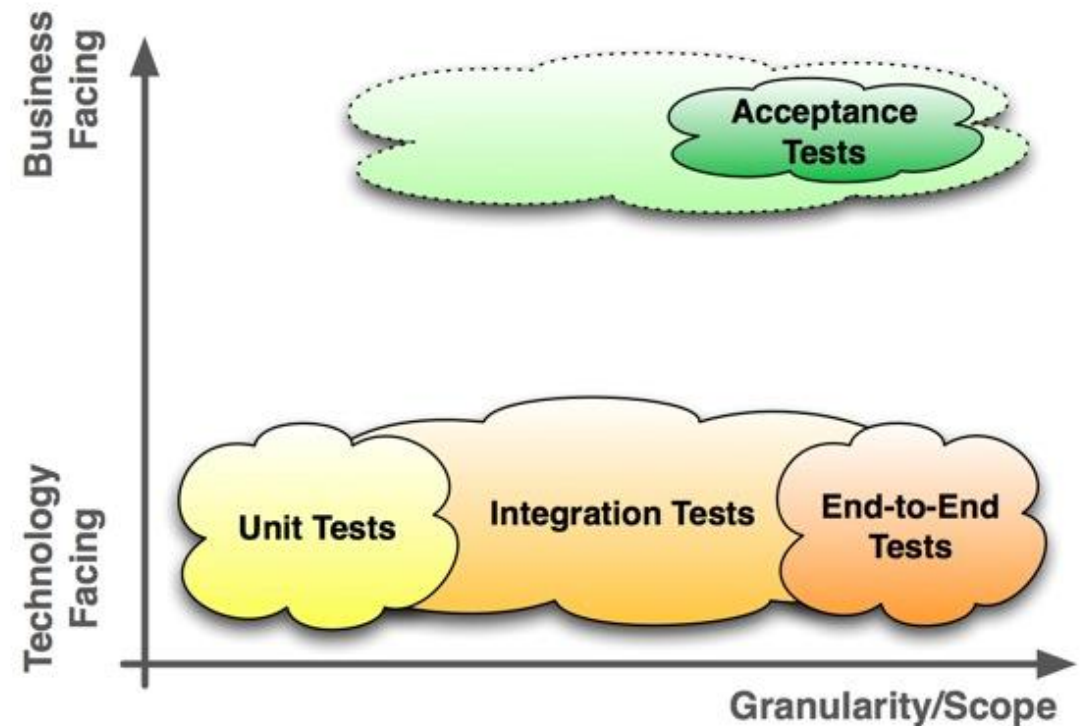
Safety Assurance

Regression Tests

Acceptance Testing

- **Three major objectives of acceptance testing:**

- Confirm that the system meets the agreed upon acceptance criteria
- Identify and resolve discrepancies, if there is any
- Determine the readiness of the system for cut-over to live operations





# TEST PLANNING

# USE-CASE TESTING – TEST CASE

<b>Test case</b>	<b>SUC3 Correct PIN entry on first try</b>
<b>Test description</b>	A customer enters the PIN number correctly on the first attempt.
<b>Related screens</b>	
<b>Pre-conditions</b>	1. The expected PIN is known 2. Screen 2 is displayed
<b>Actions</b>	1. Screen 2 shows ‘- - - -’ 2. Customer touches 1st digit 3. Screen 2 shows ‘- - - *’ 4. Customer touches 2nd digit 5. Screen 2 shows ‘- - * *’ 6. Customer touches 3rd digit 7. Screen 2 shows ‘- * * *’ 8. Customer touches 4th digit 9. Screen 2 shows ‘* * * *’ 10. Customer touches Enter 11. Screen 5 is displayed
<b>Inputs</b>	Correct PIN number: 1357
<b>Expected Outputs</b>	Select Transaction screen is active
<b>Testing environment</b>	Software interface run in Windows 10

# USE-CASE TESTING

*The use-cases developed to identify system interactions can be used as a basis for system testing.*

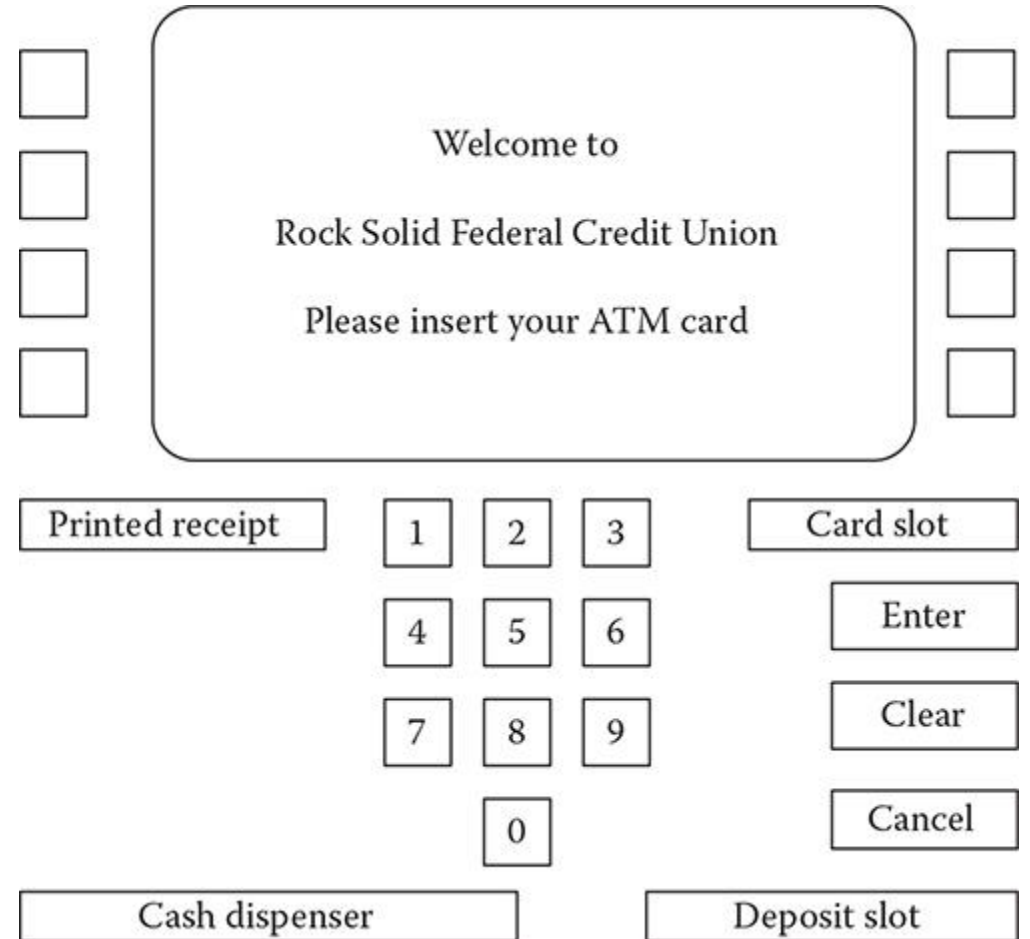
- ✓ Each use case usually involves several system components so testing the use case forces these interactions to occur.
  - The sequence diagrams associated with the use case documents the components and interactions that are being tested.

# TEST PLAN

- ✓ Testing requires meticulous planning on an operational level and on a strategic level
- ✓ The test manager needs to transpose such generic guidelines to create a concrete testing strategy for the project at hand
- ✓ Test plan should state:
  - Test object: which components, modules, neighboring systems, and interfaces (will) make up the system to be tested
  - test objectives: specific testing objectives and the criteria you are testing against for each test object and the entire system
  - Customize the testing process
  - testing methods and techniques: overall testing approach and the testing techniques
  - required infrastructure
  - Success/ failure criteria: test metrics and threshold

# A CASE — ATM TERMINAL SOFTWARE TESTING

- ✓ A simple version of ATM machine
- ✓ Insert your card, type PIN code and withdraw cash from the machine





# A CASE — ATM TERMINAL SOFTWARE

## ✓ List of designed screens

Screen 1  
Welcome  
Please insert your  
ATM card

Screen 2  
Please enter your PIN  
\_\_\_\_\_

Screen 3  
Your PIN is incorrect.  
Please try again.

Screen 4  
Invalid ATM card. It will  
be retained.

Screen 5  
Select transaction:  
balance >  
deposit >  
withdrawal >

Screen 6  
Balance is  
\$dddd.dd

Screen 7  
Enter amount.  
Withdrawals must  
be multiples of \$10

Screen 8  
Insufficient funds!  
Please enter a new  
amount

Screen 9  
Machine can only  
dispense \$10 notes

Screen 10  
Temporarily unable to  
process withdrawals.  
Another transaction?

Screen 11  
Your balance is being  
updated. Please take  
cash from dispenser.

Screen 12  
Temporarily unable to  
process deposits.  
Another transaction?

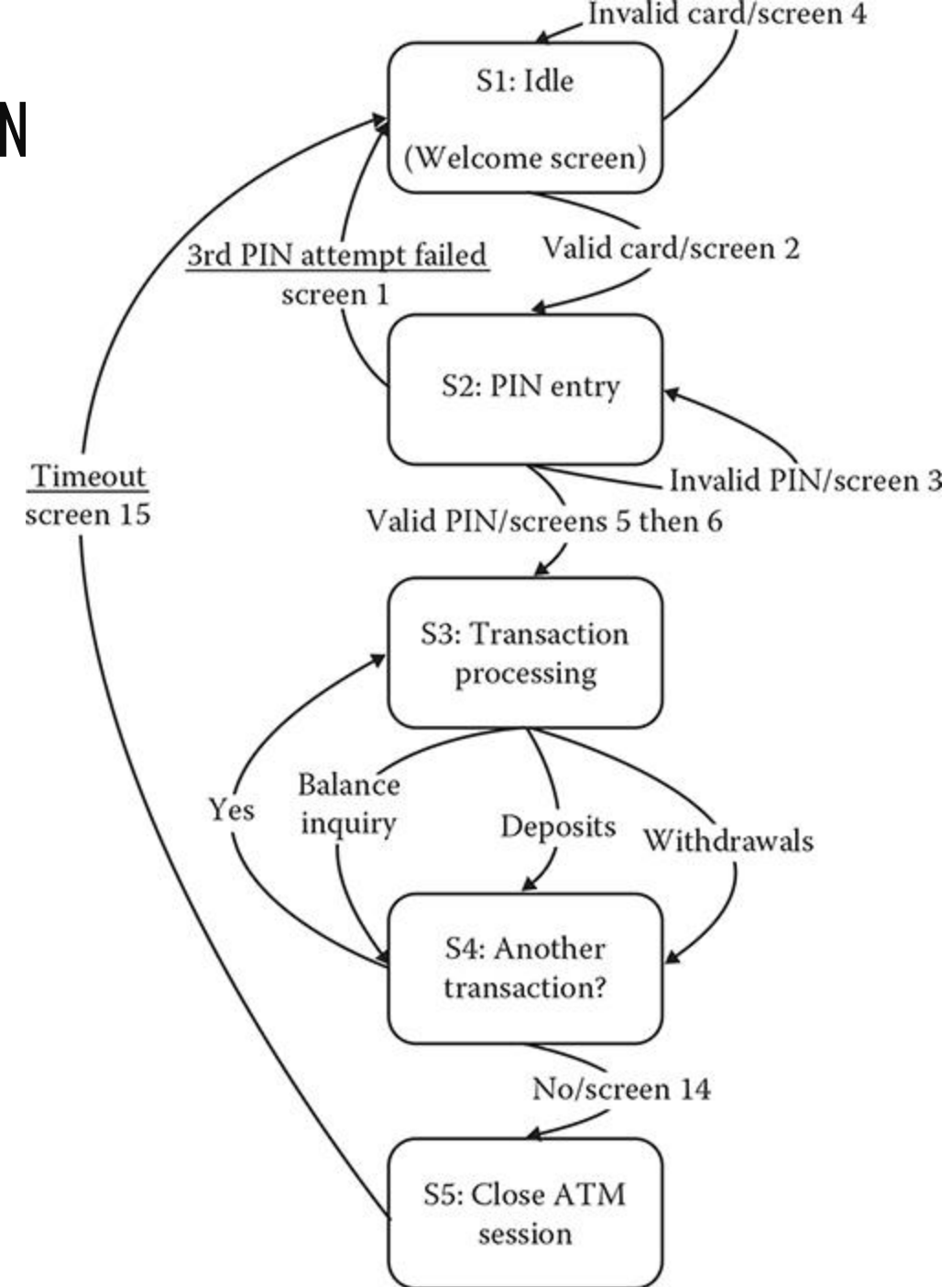
Screen 13  
Please insert deposit  
into deposit slot.

Screen 14  
Your new balance is  
being printed. Another  
transaction?

Screen 15  
Please take your  
receipt and ATM card.  
Thank you.

# A CASE — ATM TERMINAL SOFTWARE TESTIN

- ✓ State diagram from the Idle state to the Closed ATM session



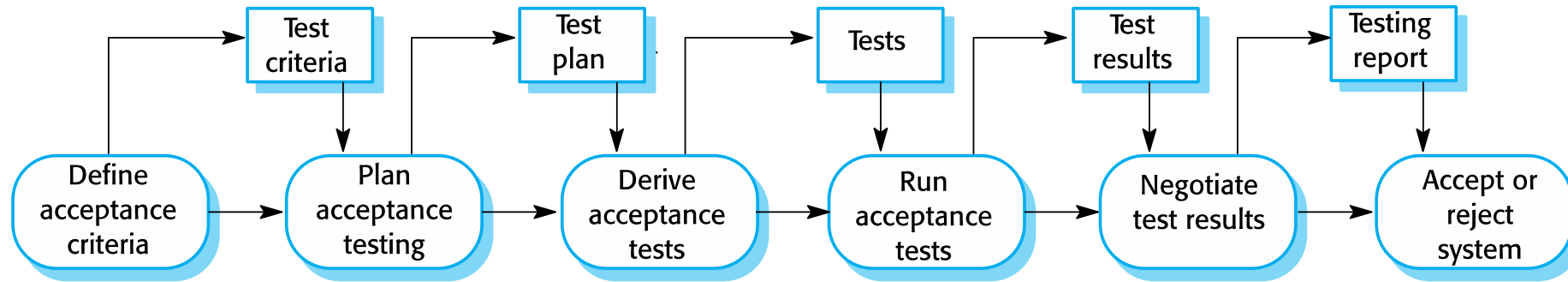
# TEST PLAN - EXAMPLE

Type of tests	Description	Number of tests	Test types	Test data	Test metric
<b>Unit test</b>	Testing each screen to make sure each screen displays according to its specification	At least 15 assertion: S1 – S15	Whitebox blackbox	Pin code, Amount to withdraw	Path coverage Line coverage Branch coverage
<b>System test/ Use case test</b>	Testing possible different use case scenarios with successful and unsuccessful ATM transactions	At least 25 test cases as below	Blackbox	Pin code, Amount to withdraw	Requirement coverage
<b>Usability test</b>	Evaluate the user experience with the ATM interfaces with the two scenarios with successful and unsuccessful ATM transactions	At least 2 test cases	Blackbox	Tasks to perform Pin code, Amount to withdraw	Usability score
<b>Performance test</b>	Simulating POS system's transaction processing under extreme conditions Internet-based transactions	1 test with 100 transactions and another test with 1000 transactions	Blackbox	Number of transactions Amount to withdraw	Transaction time Delayed time

# USER ACCEPTANCE TEST (UAT)

- ✓ acceptance testing is a test conducted to determine if the requirements of a specification or contract are met
- ✓ It can look like a system test ... with customer involvement
- ✓ It is blackbox testing
- ✓ It may involve performance test, stress test, usability test, etc
- ✓ Testers should be given real-life scenarios such as the three most common or difficult tasks that the users they represent will undertake

# STAGES IN THE ACCEPTANCE TESTING PROCESS



- ✓ Define acceptance criteria
- ✓ Plan acceptance testing
- ✓ Derive acceptance tests
- ✓ Run acceptance tests
- ✓ Negotiate test results
- ✓ Reject/accept system

# EXERCISE

Use Case: Logging into a Social Media Application

Description: The user needs to log into a social media application in order to access their account and interact with other users.

Actor: User

Precondition: The user has already installed the social media application on their device.

Basic Flow:

- ✓ The user opens the social media application.
- ✓ The application presents the login screen.
- ✓ The user enters their username and password and clicks on the "Log In" button.
- ✓ The application verifies the user's credentials and logs them into their account.
- ✓ The application presents the user's home screen with their newsfeed and other options.

Alternate Flows:

- ✓ If the user enters an incorrect username or password, the application displays an error message and prompts the user to re-enter their credentials.
- ✓ If the user forgets their password, they can click on the "Forgot Password" link and enter their email address to receive a password reset link.
- ✓ If the user does not have an account, they can click on the "Sign Up" button to create a new account.

Postcondition: The user is logged into their social media account and can access their newsfeed and interact with other users.

# EXERCISE

Use the Test plan table in Slide 59 to plan for testing this use case

Come up with at least 10 test cases for functional testing, performance testing, security testing, usability testing and compatibility testing