

EMBEDDED ACADEMY

★ PEDAL TO THE MEDAL ★



BGSV Embedded Academy (BEA)

Focused Program to Develop Embedded Competence

BGSV EMBEDDED ACADEMY

Technical Competence

T1: Automotive Basics
(Sensor, SW, Mobility
Solution)

T2: Automotive SW
Architecture (AUTOSAR)

T3: Embedded Programming

T5: Test Overview

Methodological Competence

M1: SW Development
Lifecycle

M3: Clean Code

Process Competence

P1: Requirements
Engineering

P2: Design Principles

P3: Review

P4: Safety & Security

Classroom training, Online Self-learning, Live Demo

Purpose: Develop basic general embedded competence

Disclaimer

- ▶ This slide is a part of BGSV Embedded Academy (BEA) program and only used for BEA training purposes.
- ▶ This slide is Bosch Global Software Technology Company Limited's internal property. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution as well as in the event of applications for industrial property rights.
- ▶ This slide has some copyright images and text, which belong to the respective organizations.



T5 TEST

Birthday pen selling



**This is your airbag
on time.**

Why is Testing necessary?

What do software faults cost?



THE BEA NEWS

www.BEAnews.com

THE BOSCH'S FAVOURITE NEWSPAPER

- Since 2011

Hospital Revives Its "Dead" Patients



Eighty-five hundred people at St. Mary's Mercy thought they were still alive. But the hospital's computers were telling them they were not.

In October, Cathy Uhl gave a blood sample as part of a routine physical examination, but less than two months later—according to the hospital that processed her laboratory results—she was dead.

Fortunately, Uhl and some 8,500 other patients who received treatment at St. Mary's Mercy Medical Center in Grand Rapids, Mich., between Oct. 25 and Dec. 11 were able to read of their unfortunate "demise" in the bills sent out by the hospital's flawed patient-management software system.

THE BEA NEWS

www.BEAnews.com

THE BOSCH'S FAVOURITE NEWSPAPER

- Since 2011

A Bug and a Crash



It took the European Space Agency 10 years and \$7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

All it took to explode that rocket less than a minute into its maiden voyage last June was a small computer program trying to stuff a 64-bit number into a 16-bit space.

Self-destruction was triggered automatically because aerodynamic forces were ripping the boosters from the rocket.

<http://www.around.com/ariane.html>

Why is Testing necessary?

What do software faults cost?

→ huge sums

- Ariane 5 (\$7billion)
- Mariner space probe to Venus (\$250m)
- American Airlines (\$50m)

→ very little or nothing at all

- minor inconvenience
- no visible or physical detrimental impact

→ software is not “linear”

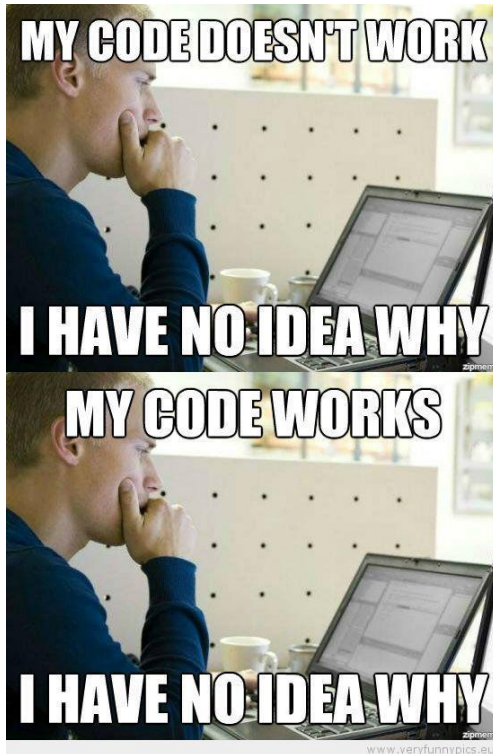
- small input may have very large effect

Why is Testing necessary?

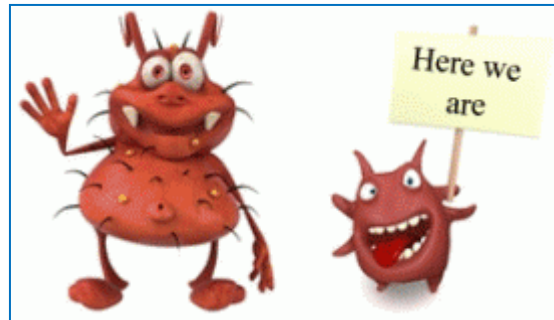


Why is Testing necessary?

A person makes
an error ...



... that creates a
fault in the
software ...



... that can cause
a failure
in operation

Why is Testing necessary?

Causes of software defects

- **Error (mistake):** a human action that produces an incorrect result
- **Fault (defect, bug):** a manifestation of an error in software
 - also known as a defect or bug
 - if executed, a fault may cause a failure
- **Failure:** deviation of the software from its expected delivery or service
 - (found defect)

A human being can make an error (mistake), which produces a defect (fault, bug) in the code, in software or a system, or in a document. If a defect in code is executed, the system will fail to do what it should do (or do something it shouldn't), causing a failure

**Failure is an event, fault is a state of
the software, caused by an error**

Why is Testing necessary?

Why do faults occur in software?

- if you have ever written software ...
- software is written by human beings
 - who know something, but not everything
 - who have skills, but aren't perfect
 - who do make mistakes (errors)
- under increasing pressure to deliver to strict deadlines
 - no time to check but assumptions may be wrong
 - systems may be incomplete

Why is Testing necessary?

Why do faults occur in software?

- Communication
- Software complexity
- Programming errors
- Changing requirements
- Poorly documented code
- Software development tools
- Environmental conditions

Why is Testing necessary?

Why not just "test everything"?

How many testcases do we need to test the below statement?

```
UWORD a_uw, b_uw;
```

```
a_uw = b_uw;
```


Why is Testing necessary?

Exhaustive testing?

→ What is exhaustive testing?

- when all the testers are exhausted ✗
- when all the planned tests have been executed ✗
- exercising all combinations of inputs and preconditions ✓

→ How much time will exhaustive testing take?

- infinite time ✗
- not much time ✗
- impractical amount of time ✓

Why is Testing necessary?

How much testing is enough?

- it's never enough ❌
- when you have done what you planned ❌
- when your customer/user is happy ❌
- when you have proved that the system works correctly ❌ ✓
- it depends on the risks for your system ✓

What is Testing?

What?

- ❖ Checking the Software is OK!!??
- ❖ The process of executing a program with the intent of **finding errors**.
- ❖ The process of executing a software system to determine whether it **matches its specification** and executes in its **intended environment**.

The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of system or component.

What is Testing?

Why?

- ❖ To check if there are any **errors** in a part or a product.
- ❖ To gain the **confidence** in the correctness of the product.
- ❖ To ensure the **quality and satisfaction** of the product.

What is Testing

- Definition as per ISTQB standard glossary
 - The process consisting of all **lifecycle activities** both **static and dynamic**, Concerned with **planning, preparation** and **evaluation of software products** and related work products to determine that **they satisfy specified requirements**, to demonstrate that they are **fit for purpose** and to **detect defects**

Objectives of testing

- Finding defects
- Gaining confidence about level of quality
- Providing information for decision making
- Preventing defects

Testing and Debugging

- Testing (done by the tester)
- Debugging (done by the developer)

V&V

V&V

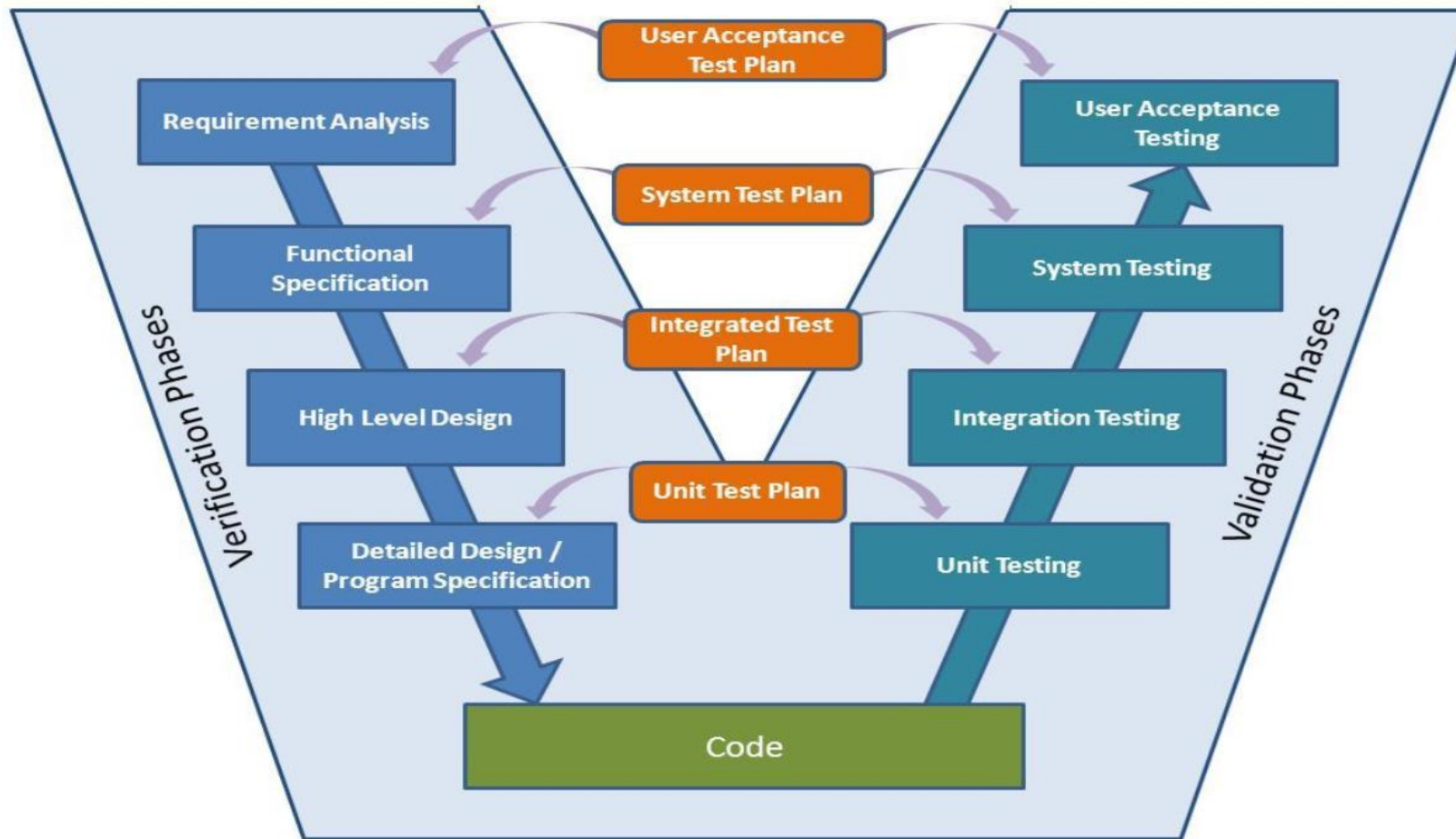
❖ **Verification** – are we building the **product correctly**?*

– The set of activities to ensure that software correctly implements specific functions

❖ **Validation** – are we building the **correct product**?*

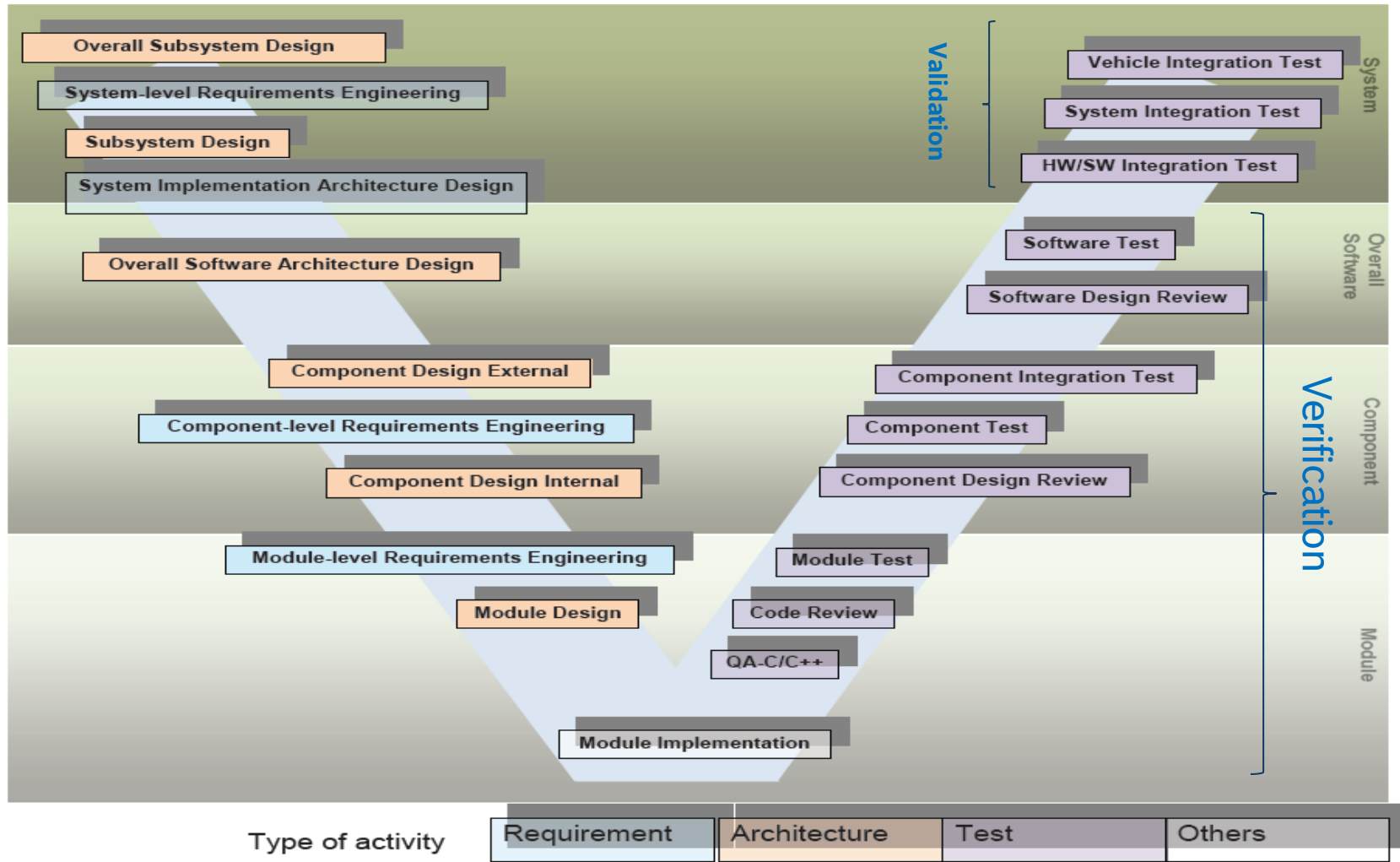
– The set of activities to ensure that the developed software is traceable to customer requirements

V Model



Source: <http://crackmba.com/v-shaped-model/>

V&V Model



Seven Testing Principles

Principle 1 – Testing shows presence of defects

❖ Testing can show the presence of defects, but cannot prove there are no defects.

Principle 2 – Exhaustive testing is impossible

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases

Principle 3 – Early testing

Testing early as possible

Principle 4 – Defect clustering

Most of defects are found in a small number of module

Principle 5 – Pesticide paradox

Test cases also need to be updated

Principle 6 – Testing is context dependent

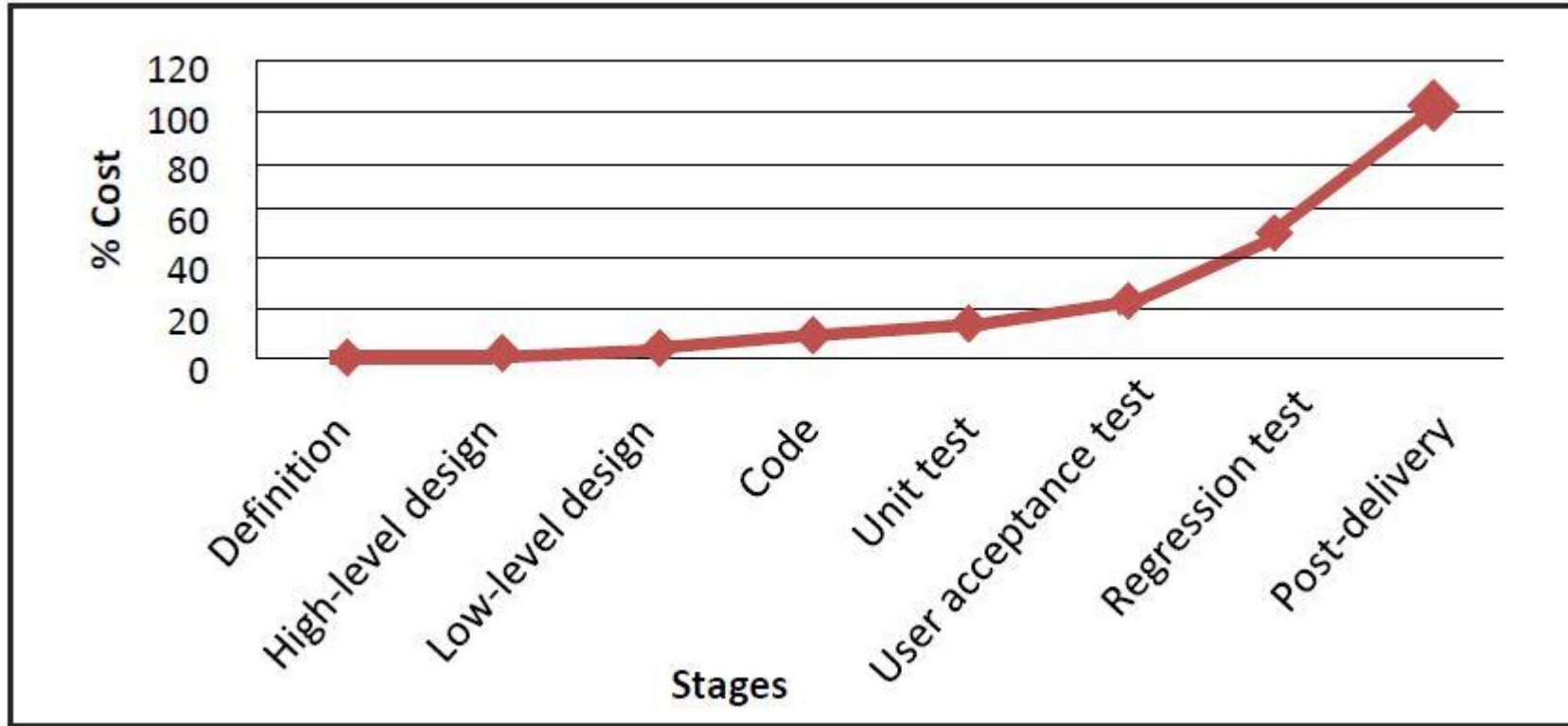
Testing is done differently in different contexts.

Principle 7 – Absence-of-errors fallacy

Testing is useless if the system does not meet the user's requirements

Prioritise tests so that, whenever you stop testing, you have done the best testing in the time available.

Cost of fixing faults



Source: <https://hkrtrainings.com/istqb-tutorial>

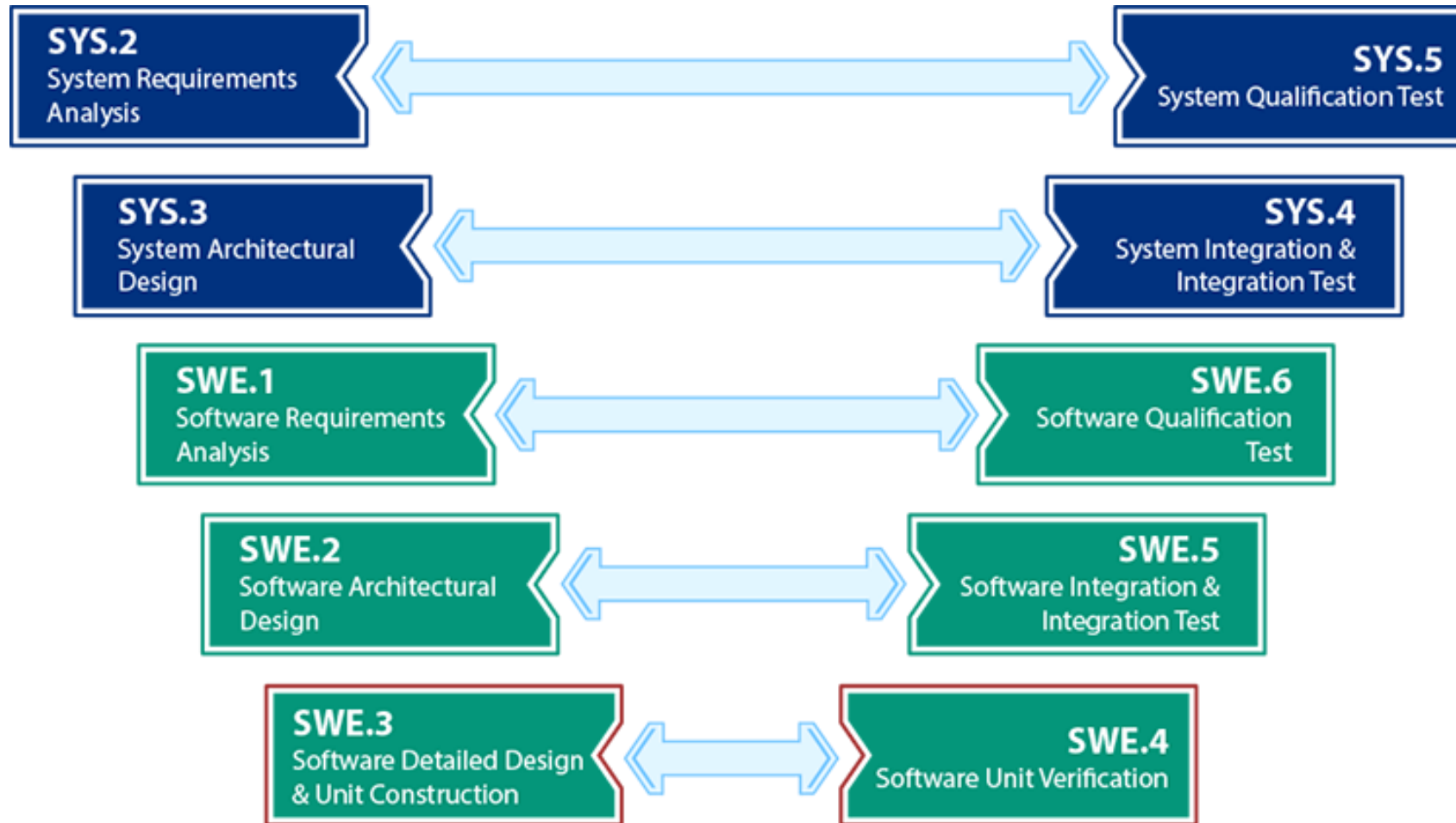
Cost of fixing faults

Software Testing Phase Where Bugs Were Found	Estimated Cost per Bug
System Testing	\$5,000
Integration Testing	\$500
Full Build	\$50
Unit Testing/Test-Driven Development	\$5

Table 2: The cost to fix bugs at Google

<https://www.coderskitchen.com/cost-of-fixing-vs-preventing-bugs/>

V Model in ASPICE

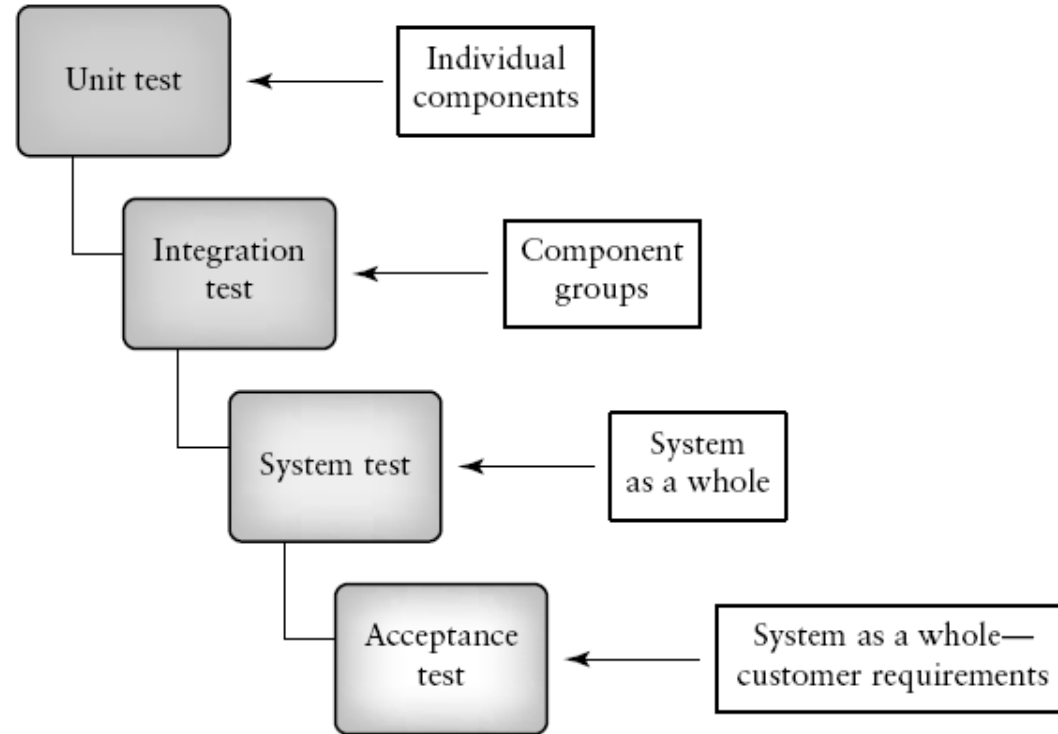


Source: <https://evav.omnex.com/7-levers/automotive-spice>

Test Levels

Test Levels

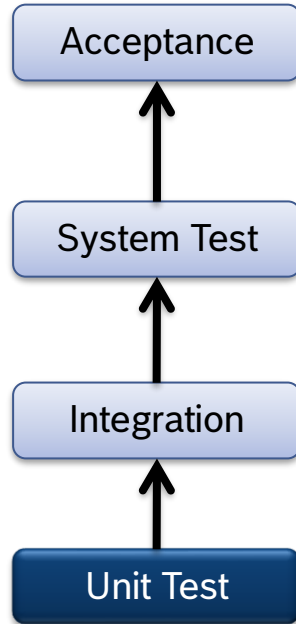
- Unit testing
- Integration testing
- System testing
- Acceptance testing



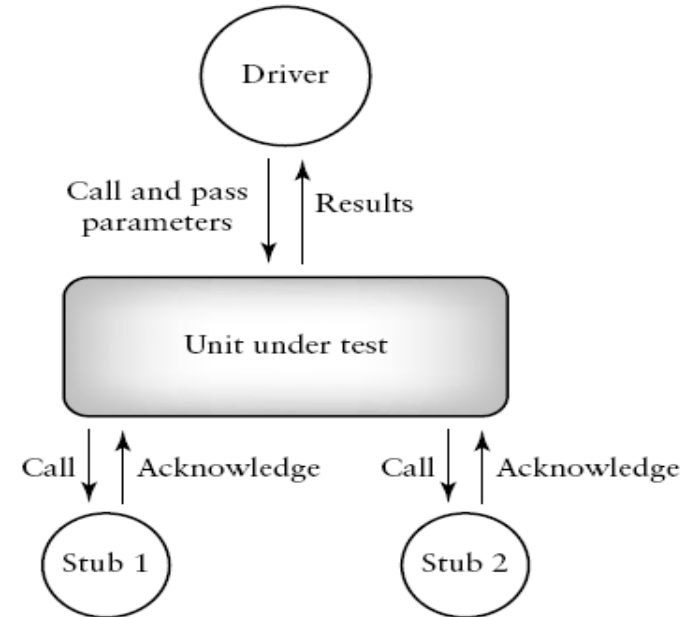
Courtesy : Practical Software Testing , illene Burnstein

Test Levels

Unit Test

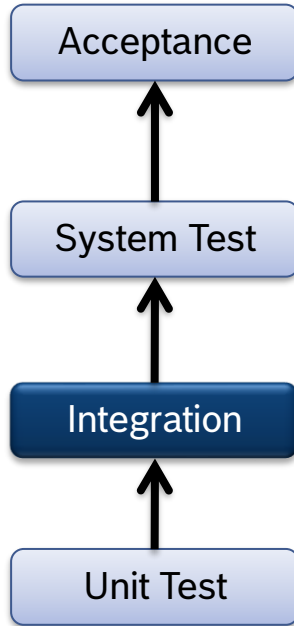


- ❖ A minimal software item for which a separate specification is available.
- ❖ lowest level
- ❖ tested in isolation
- ❖ Separately testable
- ❖ usually done by programmer
- ❖ Stubs, Drivers and Simulators
- ❖ also known as component, module, program testing



Test Levels

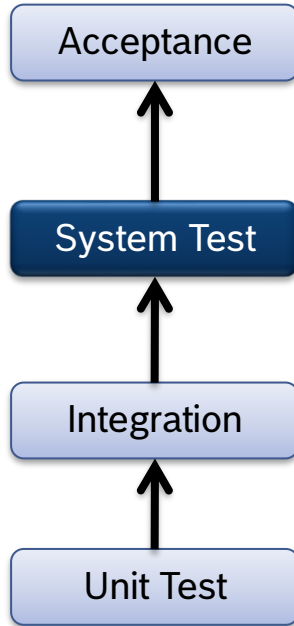
Integration Test



- ❖ more than one (tested) component
- ❖ communication between components
- ❖ integration strategy: big-bang vs incremental (top-down, bottom-up, functional)
- ❖ done by designers, analysts, or independent testers
- ❖ test non-functional aspects such as performance

Test Levels

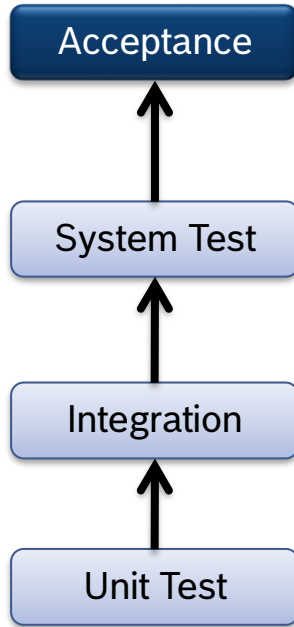
System Test



- ❖ last integration step
- ❖ functional
 - ❖ functional requirements and requirements-based testing
 - ❖ business process-based testing
- ❖ non-functional
 - ❖ as important as functional requirements
 - ❖ often poorly specified
 - ❖ must be tested
- ❖ often done by independent test group

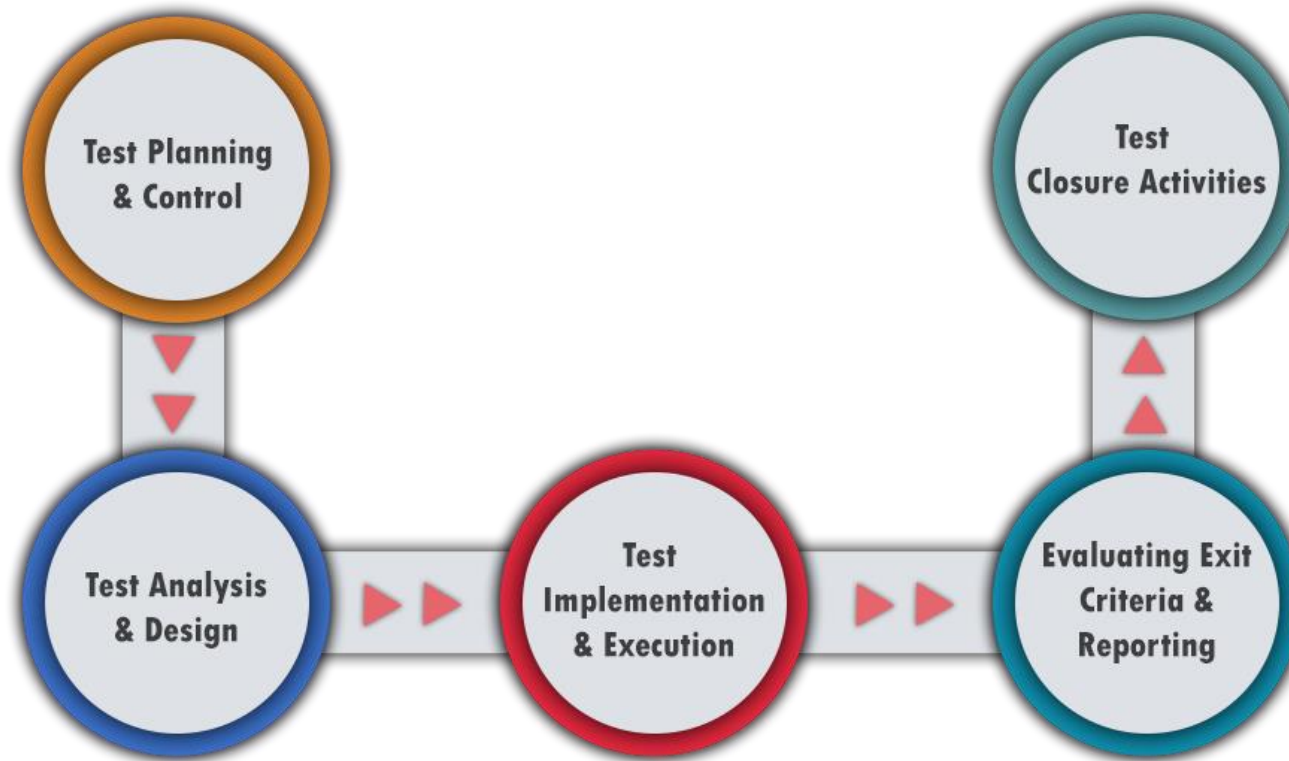
Test Levels

Acceptance Test



- ❖ Final stage of validation
- ❖ customer (user) should perform or be closely involved
- ❖ customer can perform any test they wish, usually based on their business processes
- ❖ Use the product in a realistic way in its operational environment
- ❖ Focuses on building confidence rather than finding faults
- ❖ Alpha and Beta testing

Fundamental Test Process



Source: <https://hkrtrainings.com/istqb-tutorial>

ISTQB® - FOUNDATION LEVEL

Fundamentals of Testing	Testing Throughout the Software Development Lifecycle	Static Testing	Test Techniques	Test Management	Tool Support for Testing
What is Testing?	Software Development Lifecycle Models	Static Testing Basics	Categories of Test Techniques	Test Organisation	Test Tool Considerations
Why is Testing Necessary?	Test Levels	Review Process	Black-box Test Techniques	Test Planning and Estimation	Effective Use of Tools
Seven Testing Principles	Test Types		White-box Test Techniques	Test Monitoring and Control	
Test Process	Maintenance Testing		Experience-based Test Techniques	Configuration Management	
The Psychology of Testing				Risk and Testing	
				Defect Management	



shutterstock.com • 1831072243

Why need test techniques?

- Exhaustive testing (use of all possible inputs and conditions) is impractical
 - must use a subset of all possible test cases
 - must have high probability of detecting faults
- Need thought processes that help us select test cases more intelligently
 - test case design techniques are such thought processes

What is a testing technique?

- a procedure for selecting or designing tests
- based on a structural or functional model of the software
- successful at finding faults
- 'best' practice
- a way of deriving good test cases
- a way of objectively measuring a test effort

Three types of systematic technique

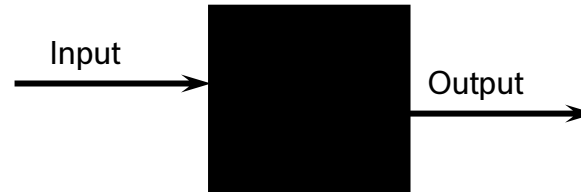
Static (non-execution)

- ❖ examination of documentation, source code listings, etc.



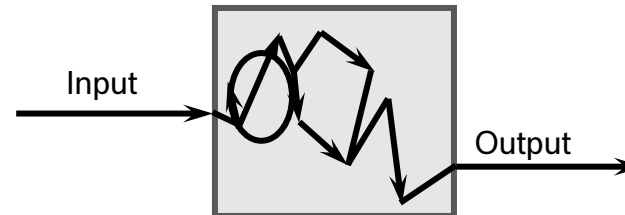
Functional (Black Box)

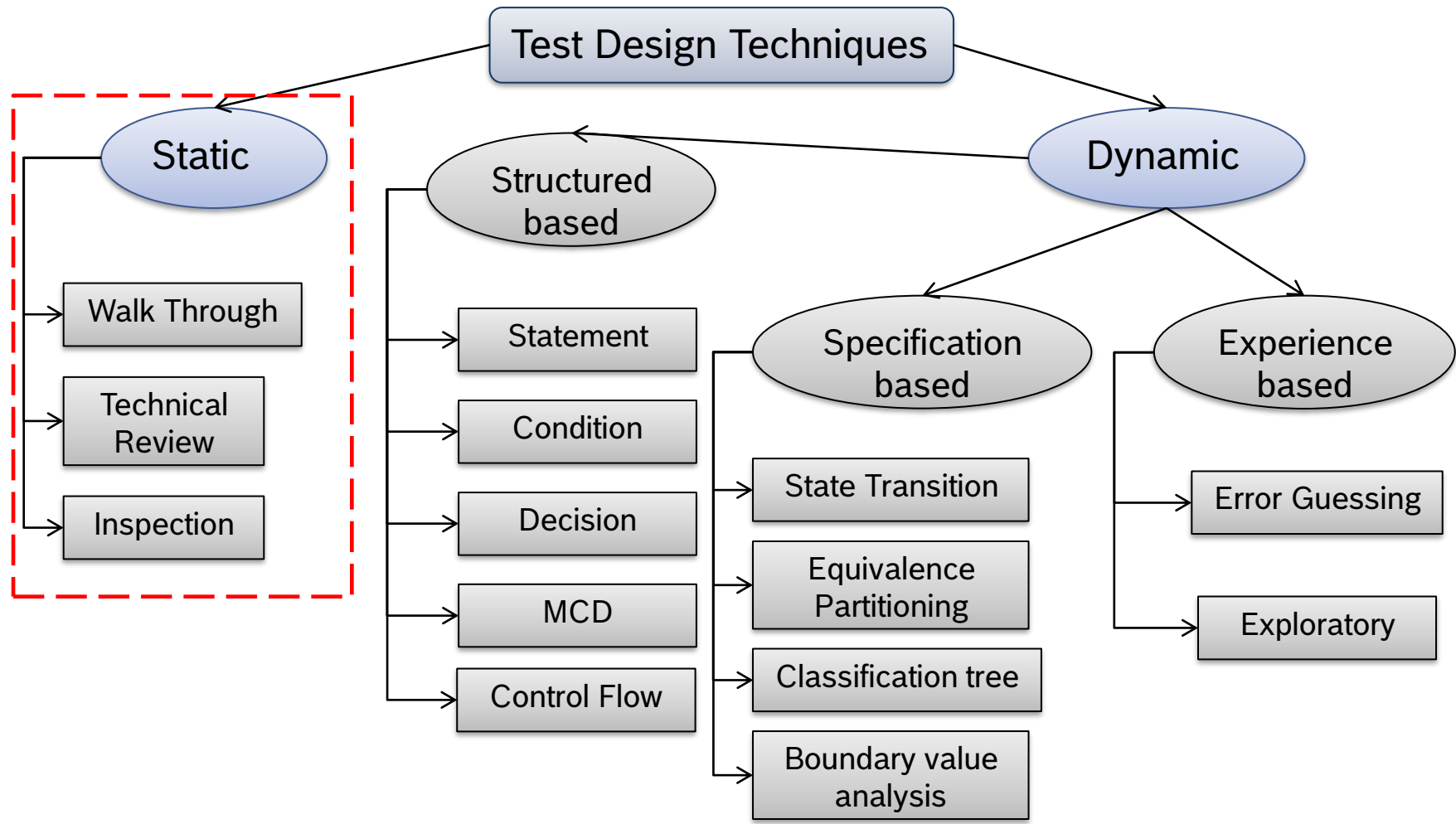
- ❖ based on behaviour / functionality of software



Structural (White Box)

- ❖ based on structure of software





Static techniques do not execute code!!

- To be used before any tests are executed on the software
- Can be used to “test” any form of document including source code, design document and models, functional specifications and requirement specifications

Benefits:

- ➔ Start early in life cycle, early feedback on quality issues can be established
- ➔ By detecting defects at an early stage, rework costs are most often low
- ➔ Rework effort is reduced, development productivity figures are likely to increase

Informal review

widely viewed as useful and cheap (but no one can prove it!) A helpful first step for chaotic organisations.

Technical review or Peer review

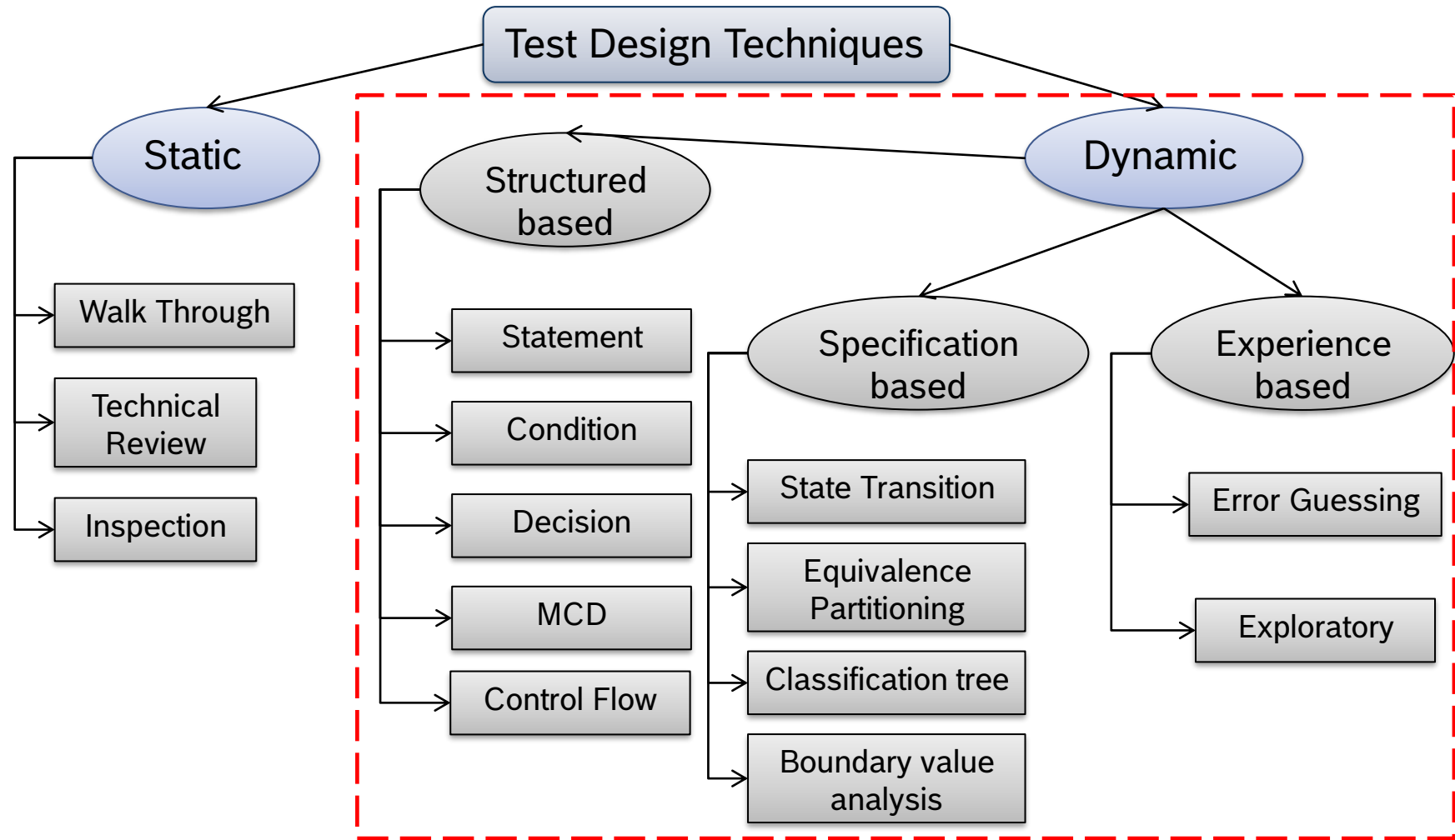
includes peer and technical experts, no management participation. Normally documented, fault-finding. Can be rather subjective

Walkthrough

author guides the group through a document and his or her thought processes, so all understand the same thing, consensus on changes to make

Inspection

formal individual and group checking, using sources and standards, according to generic and specific rules and checklists, using entry and exit criteria, Leader must be trained & certified, metrics required



Specification-based (black-box) testing techniques

A procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional of a component or system without reference to its internal structure.

- View the software as a black-box with inputs and outputs
- Have no knowledge of how the system or component is structured inside the box

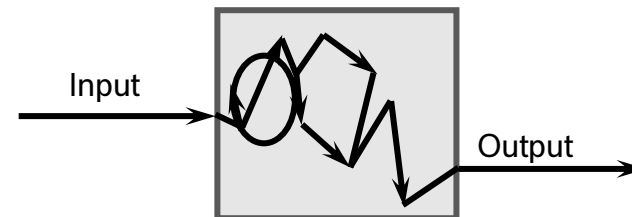


- Concentrate on what the software does, not how it does
- Are appropriate at all levels of testing where a specification exists

Structure-based (white-box) testing techniques

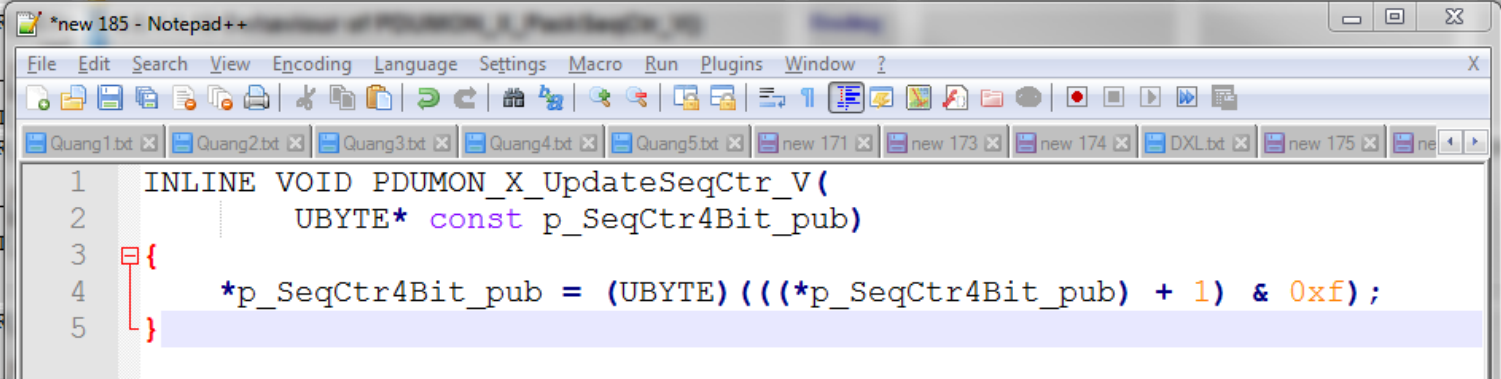
A procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system

- Use the internal structure of the software to derive test cases
- Require knowledge of how the software is implemented, that is, how it works
- Can be used at all levels of testing, especially where there is good tool support for code coverage

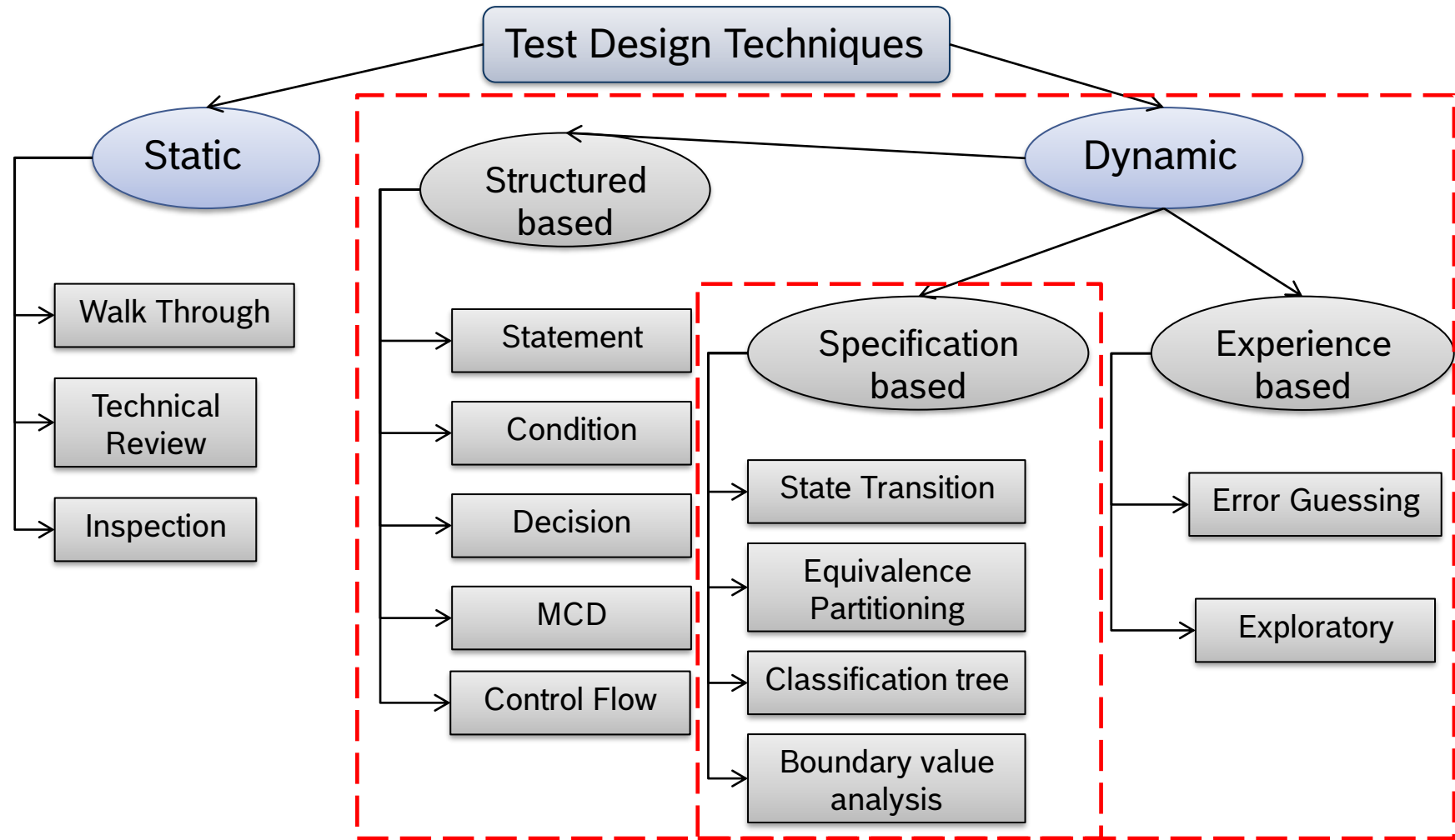


White box or Black box testing techniques?

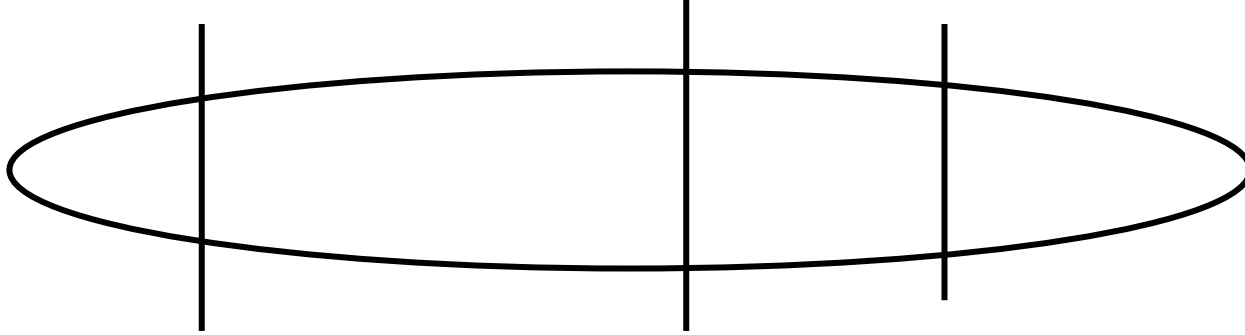
ID	SW Modules Requirements Specifications	Object Type	Status_MLBevo_D5_1_0	Release_D5	Rel
R_MLBevo2_SWM_SIO_NE_T_R_431	4.6.1.1.12 Behaviour of PDUMON_X_UpdateSeqCtr_V()	Heading			
R_MLBevo2_SWM_SIO_NE_T_R_432	<i>PDUMON_X_UpdateSeqCtr_V()</i> shall increment the value of a referenced four bit sequence counter.	Requirement	Implemented		X0
R_MLBevo2_SWM_SIO_NE_T_R_433	If the maximal value "15" already was reached <i>PDUMON_X_UpdateSeqCtr_V()</i> shall continue with "0".	Requirement	Implemented		X0



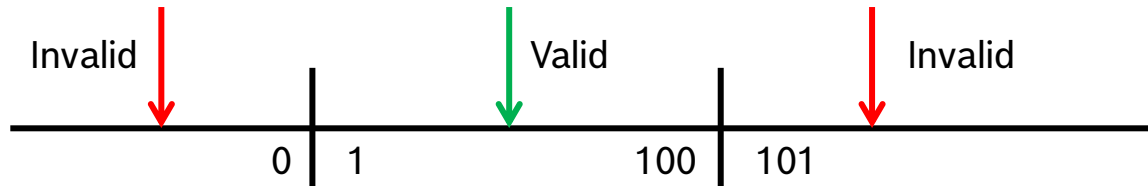
```
1  INLINE VOID PDUMON_X_UpdateSeqCtr_V(  
2      UBYTE* const p_SeqCtr4Bit_pub)  
3  {  
4      *p_SeqCtr4Bit_pub = (UBYTE) (((*p_SeqCtr4Bit_pub) + 1) & 0xf);  
5  }
```



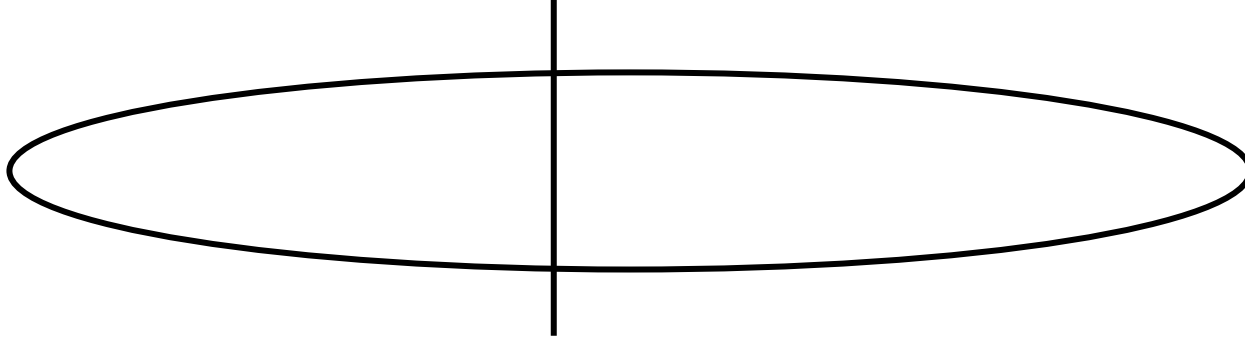
Equivalence partitioning (EP)



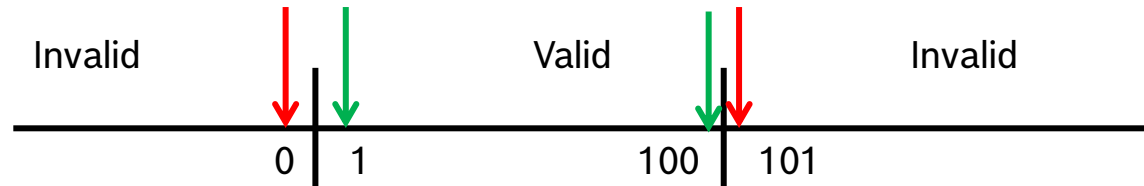
- Divide (partition) the inputs, outputs, etc. into areas which are the same (equivalent)
- Assumption: if one value works, all will work
- One from each partition better than all from one



Boundary Value Analysis (BVA)



- ❑ Faults tend to be found near boundaries
- ❑ Good place to look for faults
- ❑ Test values on both sides of boundaries



Why do both EP and BVA?

- If you do boundaries only, you have covered all the partitions as well
 - technically correct and may be OK if everything works correctly!
 - if the test fails, is the whole partition wrong, or is a boundary in the wrong place - have to test mid-partition anyway
 - testing only extremes may not give confidence for typical use scenarios (especially for users)
 - boundaries may be harder (more costly) to set up

What are the equivalence partitioning & boundary values of below function?

```
1 char FuncA(char x, char y)
2 {
3     if ((x>=10) && (x<100))
4     {
5         y = 1;
6     }
7     else if ((x>=100) && (x<200))
8     {
9         y = 2;
10    }
11    else if (x>=200)
12    {
13        y = 3;
14    }
15    return y;
16 }
```

What are the equivalence partitioning & boundary values of below function?

```
1  unsigned char FuncA(unsigned char x, unsigned char y)
2  {
3      if ((x>=10) && (x<100))
4      {
5          y = 1;
6      }
7      else if ((x>=100) && (x<200))
8      {
9          y = 2;
10     }
11     else if (x>=200)
12     {
13         y = 3;
14     }
15     return y;
16 }
```

Thank you!

Bosch
Global
Software
Technologies
alt_future