STQM

UNIT-1

Testing-Introduction

- What is software testing?
- Why do we need to test software?
- Can we live without testing?
- How do we handle software bugs reported by the customers?
- Who should work hard to avoid frustrations of software failures?

SOME SOFTWARE FAILURES

- Ariane 5 rocked exploded in June 1996 36 seconds after it was launched.
- Reason: Software error
- Exception occurred during conversion of a 64-bit floating point number into a 16-bit integer, backup software also failed due to same reason.
- Rocket failed due to incorrect data transmission regarding altitude.

Reference-[1],[4],[5]

- Six patients died due to an overdose of radiation caused from Therac 25, a medical linear accelerator that is used to treat tumors.
- The main cause of error was a race condition caused by wrong sequence of commands caused by the software operating the accelerator.

Reference-[1],[4],[5]

- Intel lost an estimated $475 million due to a defective pentium chip.
- The chip made mistakes while dividing floating point numbers within a certain range.
- For example, 3145727 x 4195835 / 3145727 should return 4195835. A flawed Pentium will return 4195579!
- Intel had to replace most of 3 to 5 million defective chips in circulation.

Reference-[1],[4],[5]

Testing

- Agile methodologies insist on developers unit testing their code thoroughly. Testing is not a testers job alone.
- Embedded safety critical, control software has to be tested with extra care to meet regulatory requirements.
- Enterprise software is very complex— large data bases, critical server requirements etc.
- Web applications are available to more users, need to be correct.
- Free software is also expected to be correct!

Reference-[1],[4],[5]

TESTING PROCESS

- Process of testing the newly developed software, prior to its actual use.
- The program is executed with desired input(s) and the output(s) is/are observed accordingly.
- The observed output(s) is/are compared with expected output(s).
- If both are same, then the program is said to be correct as per specifications, otherwise there is something wrong somewhere in the program. T

Reference-[1],[4],[5]

TESTING PROCESS

- Testing is a very expensive process and consumes one-third to one-half of the cost of a typical development project.
- It is largely a systematic process but partly intuitive too.
- Hence, good testing process entails much more than just executing a program a few times to see its correctness.

Reference-[1],[4],[5]

What is Software Testing?

- Good testing entails more than just executing a program with desired input(s).

Reference-[1],[4],[5]

```
1.  void Minimum();
2.  void main()
3.  {
4.          Minimum();
5.  }
6.  void Minimum()
7.  {
8.          int array[100];
9.          int Number;
10.         int i;
11.         int tmpData;
12.         int Minimum=INT_MAX;
13.         clrscr();
14.         "printf("Enter the size of the array:");
15.         scanf("%d",&Number);
16.         for(i=0;i<Number;i++) {
17.                 printf("Enter A[%d]=",i+1);
18.                 scanf("%d",&tmpData);
19.                 tmpData=(tmpData<0)?-tmpData:tmpData;
20.                 array[i]=tmpData;
21.         }
22.         i=1;
23.         while(i<Number-1) {
24.                 if(Minimum>array[i])
25.                 {
26.                         Minimum=array[i];
27.                 }
28.                 i++;
29.         }
30.         printf("Minimum = %d\n", Minimum);
31.         getch();
32. }
```

Reference-[1],[4],[5]

Test Case	Inputs		Expected Output	Observed Output	Match?
	Size	Set of Integers			
1.	5	6, 9, 2, 16, 19	2	2	Yes
2.	7	96, 11, 32, 9, 39, 99, 91	9	9	Yes
3.	7	31, 36, 42, 16, 65, 76, 81	16	16	Yes
4.	6	28, 21, 36, 31, 30, 38	21	21	Yes
5.	6	106, 109, 88, 111, 114, 116	88	88	Yes
6.	6	61, 69, 99, 31, 21, 69	21	21	Yes
7.	4	6, 2, 9, 5	2	2	Yes
8.	4	99, 21, 7, 49	7	7	Yes

Reference-[1],[4],[5]

Definitions of testing

- (i) Testing is the process of demonstrating that errors are not present.
- (ii) The purpose of testing is to show that a program performs its intended functions correctly.
- (iii) Testing is the process of establishing confidence that a program does what it is supposed to do.

Testing

- Our objective should not be to show that the program works as per specifications
- . But, we should do testing with the assumption that there are faults and our aim should be to remove these faults at the earliest.
- Thus, a more appropriate definition is [MYER04]: “Testing is the process of executing a program with the intent of finding faults.”

Reference-[1],[4],[5]

Testing

- If our objective is to show that a program has no errors, then we shall sub-consciously work towards this objective. We shall intend to choose those inputs that have a low probability of making a program fail.
- if our objective is to show that a program has errors, we may select those test cases which have a higher probability of finding errors. We shall focus on weak and critical portions of the program to find more errors.

Reference-[1],[4],[5]

- We shall focus on weak and critical portions of the program to find more errors.
- This type of testing will be more useful and meaningful.

Reference-[1],[4],[5]

- (i) A very short list (of inputs) with the size of 1, 2, or 3 elements.
- (ii) An empty list i.e. of size 0.
- (iii) A list where the minimum element is the first or last element.
- (iv) A list where the minimum element is negative.
- (v) A list where all elements are negative.
- (vi) A list where some elements are real numbers.
- (vii) A list where some elements are alphabetic characters.
- (viii) A list with duplicate elements.
- (ix) A list where one element has a value greater than the maximum permissible limit of an integer.

Software testing

- find many similar situations which may be very challenging and risky for this program and each such situation should be tested separately.

Reference-[1],[4],[5]

S. No.	Possible Reasons
Case 1	
A very short list with size 1, 2 or 3	While finding the minimum, the base value of the index and/or end value of the index of the usable array has not been handled properly (see line numbers 22 and 23).
Case 2	
An empty list i.e. of size 0	The program proceeds without checking the size of the array (see line numbers 15 and 16).
Case 3	
A list where the minimum element is the first or last element	Same as for Case 1.
Case 4	
A list where the minimum element is negative	The program converts all negative integers into positive integers (see line number 19).
Case 5	
A list where all elements are negative	Same as for Case 4.

S. No.	Possible Reasons
Case 6 A list where some elements are real numbers	The program uses scanf() function to read the values. The scanf() has unpredictable behaviour for inputs not according to the specified format. (See line numbers 15 and 18).
Case 7 A list where some elements are alphabetic characters	Same as for Case 6.
Case 8 A list with duplicate elements	(a) Same as for Case 1. (b) We are getting the correct result because the minimum value is in the middle of the list and all values are positive.
Case 9 A list with one value greater than the maximum permissible limit of an integer	This is a hardware dependent problem. This is the case of the overflow of maximum permissible value of the integer. In this example, 32 bits integers are used.

Changes in the program

- (i) The program has ignored the first and last values of the list The program is not handling the first and last values of the list properly. If we see the line numbers 22 and 23 of the program, we will identify the causes. There are two faults. Line number 22 “i = 1;” should be changed to “i = 0;” in order to handle the first value of the list. Line number 23 “while (i<Number -1)” should be changed to “while (i<=Number-1)” in order to handle the last value of the list.

- (ii) The program proceeds without checking the size of the array If we see line numbers 14 and 15 of the program, we will come to know that the program is not checking the size of the array / list before searching for the minimum value. A list cannot be of zero or negative size. If the user enters a negative or zero value of size or value greater than the size of the array, an appropriate message should be displayed. Hence after line number 15, the value of the size should be checked as under:

```
if (Number < = 0||Number>100)
{printf ("Invalid size specified");}
```

Reference-[1],[4],[5]

- (iii) Program has converted negative values to positive values Line number 19 is converting all negative values to positive values. That is why the program is not able to handle negative values. We should delete this line to remove this fault.

Modified Program

```
10.                     int i;
11.                     int tmpData;
12.                     int Minimum=INT_MAX;
13.                     clrscr();
14.                     printf("Enter the size of the array:");
15.                     scanf("%d",&Number);
16.                     if(Number<=0||Number>100) {
17.                     printf("Invalid size specified");
18.                     }
19.                     else {
20.                     for(i=0;i<Number;i++) {
21.                             printf("Enter A[%d]=",i+1);
22.                             scanf("%d",&tmpData);
23.                             /*tmpData=(tmpData<0)?-tmpData:tmpData;*/
24.                             array[i]=tmpData;
25.                     }
26.                     i=0;
27.                     while(i<=Number-1) {
28.                             if(Minimum>array[i])
29.                             {
30.                                     Minimum=array[i];
31.                             }
32.                             i++;
33.                     }
34.                     printf("Minimum = %d\n", Minimum);
35.                     }
36.                     getch();
37.             }
```

Why Should We Test?

Why Should We Test?

- Software testing is a very expensive and critical activity; but releasing the software without testing is definitely more expensive and dangerous.
- Hence testing is essential; but how much testing is required? Do we have methods to measure it? Do we have techniques to quantify it?
- All projects are different in nature and functionalities and a single yardstick may not be helpful in all situations. I

Reference-[1],[4],[5]

Why Should We Test?

- The basic issue of this discussion is that we cannot release a software system without adequate testing
- When to release the software is a very important decision. Economics generally plays an important role.
- We shall try to find more errors in the early phases of software development. The cost of removal of such errors will be very reasonable as compared to those errors which we may find in the later phases of software development.
- The cost to fix errors increases drastically from the specification phase to the test phase and finally to the maintenance phase

Reference-[1],[4],[5]

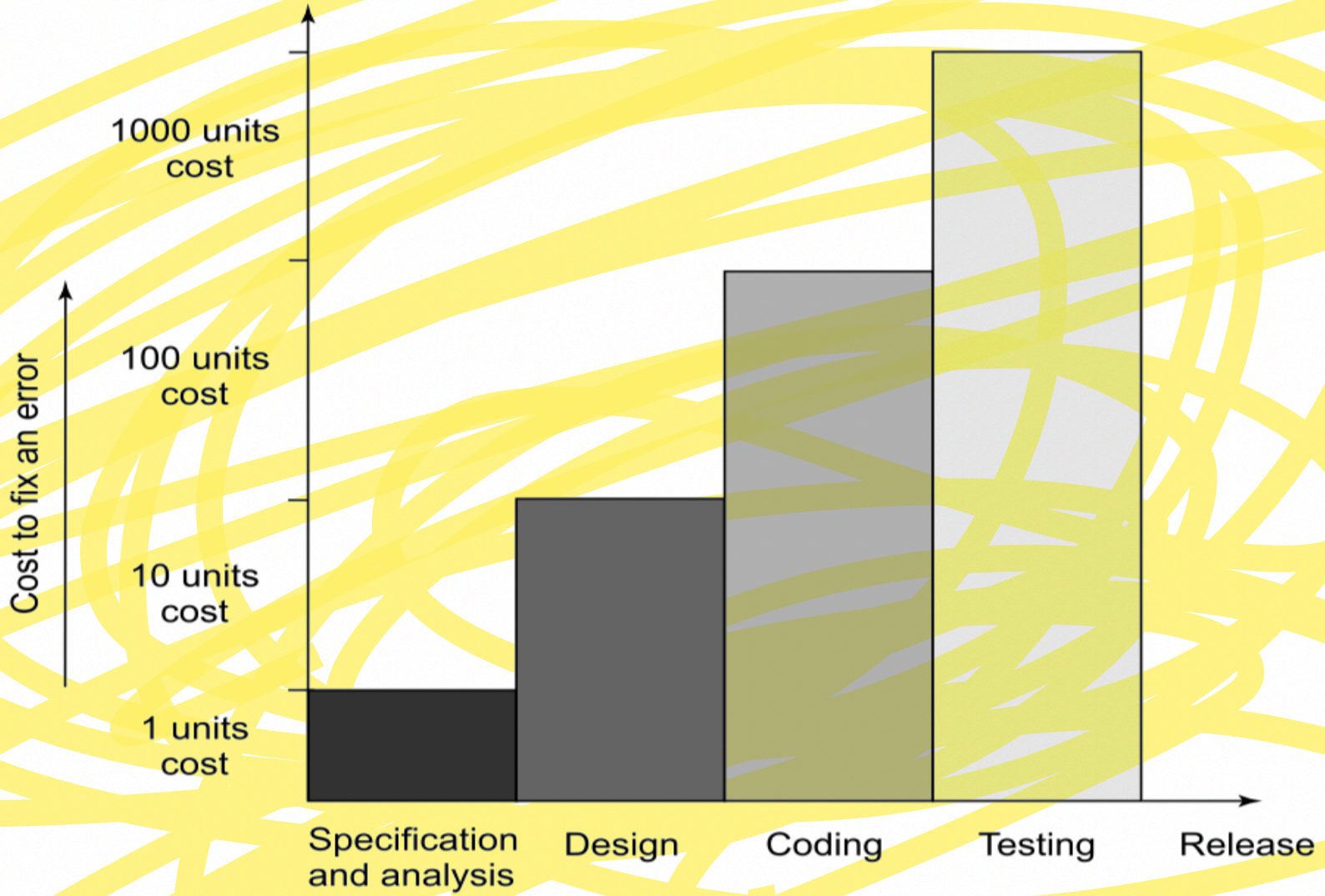
Why Should We Test?

- If an error is found and fixed in the specification and analysis phase, it hardly costs anything. We may term this as '1 unit of cost' for fixing an error during specifications and analysis phase.
- The same error, if propagated to design, may cost 10 units and if, further propagated to coding, may cost 100 units. If it is detected and fixed during the testing phase, it may lead to 1000 units of cost.
- If it could not be detected even during testing and is found by the customer after release, the cost becomes very high.

Reference-[1],[4],[5]

Why Should We Test?

- The fact is that we are releasing software that is full of errors, even after doing sufficient testing. No software would ever be released by its developers if they are asked to certify that the software is free of errors.
- Testing, therefore, continues to the point where it is considered that the cost of testing processes significantly outweighs the returns.



Reference-[1],[4],[5]

Who Should We Do the Testing?

- Testing a software system may not be the responsibility of a single person.
- It is a team work and the size of the team is dependent on the complexity, criticality and functionality of the software under test.
- The software developers should have a reduced role in testing, if possible.
- The concern here is that the developers are intimately involved with the development of the software and thus it is very difficult for them to point out errors from their own creations.

Reference-[1],[4],[5]

Who Should We Do the Testing?

- The testing persons must be cautious, curious, critical but non-judgmental and good communicators.
- One part of their job is to ask questions that the developers might not be able to ask themselves or are awkward, irritating, insulting or even threatening to the developers.

Reference-[1],[4],[5]

Who Should We Do the Testing?

- (i) How is the software?
- (ii) How good is it?
- (iii) How do you know that it works? What evidence do you have?
- (iv) What are the critical areas?
- (v) What are the weak areas and why?
- (vi) What are serious design issues?
- (vii) What do you feel about the complexity of the source code?

Reference-[1],[4],[5]

Who Should We Do the Testing?

Table 1.6. Persons and their roles during development and testing

S. No.	Persons	Roles
1.	Customer	Provides funding, gives requirements, approves changes and some test results.
2.	Project Manager	Plans and manages the project.
3.	Software Developer(s)	Designs, codes and builds the software; participates in source code reviews and testing; fixes bugs, defects and shortcomings.
4.	Testing co-ordinator(s)	Creates test plans and test specifications based on the require-ments and functional and technical documents.
5.	Testing person(s)	Executes the tests and documents results.

Reference-[1],[4],[5]

What Should We Test?

- Is it possible to test the program for all possible valid and invalid inputs?

Reference-[1],[4],[5]

What Should We Test?

- We consider a simple example where a program has two 8 bit integers as inputs. Total combinations of inputs.
- If only one second is required (possible only with automated testing) to execute one set of inputs, it may take 18 hours to test all possible combinations of inputs.

Reference-[1],[4],[5]

What Should We Test?

- Inputs are more than two and the size is also more than 8 bits. What will happen when inputs are real and imaginary numbers?
- We may wish to go for complete testing of the program, which is neither feasible nor possible.
- This situation has made this area very challenging where the million dollar question is, "How to choose a reasonable number of test cases out of a large pool of test cases?
- " Researchers are working very hard to find the answer to this question. Many testing techniques attempt to provide answers to this question in their own ways. However, we do not have a standard yardstick for the selection of test cases.

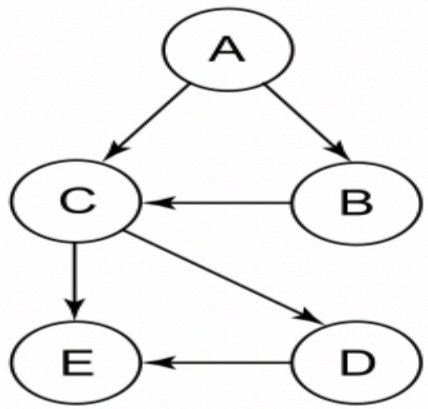
Reference-[1],[4],[5]

What Should We Test?

- (i) Execute every statement of the program at least once.
- (ii) Execute all possible paths of the program at least once.
- (iii) Execute every exit of the branch statement at least once.

```
1. if (x > 0)
2. {
3. a = a + b;
4. }
5. if (y>10)
6. {
7. c=c+d;
8. }
```

This code can be represented graphically as:



Line Numbers	Symbol for representation
1	A
2, 3, 4	B
5	C
6, 7, 8	D
End	E

Reference-[1],[4],[5]

What Should We Test?

- The possible paths are: ACE, ABCE, ACDE and ABCDE. However, if we choose x = 9 and y = 15, all statements are covered. Hence only one test case is sufficient for 100% statement coverage by traversing only one path ABCDE.

- Therefore, 100% statement coverage may not be sufficient, even though that may be difficult to achieve in real life programs.

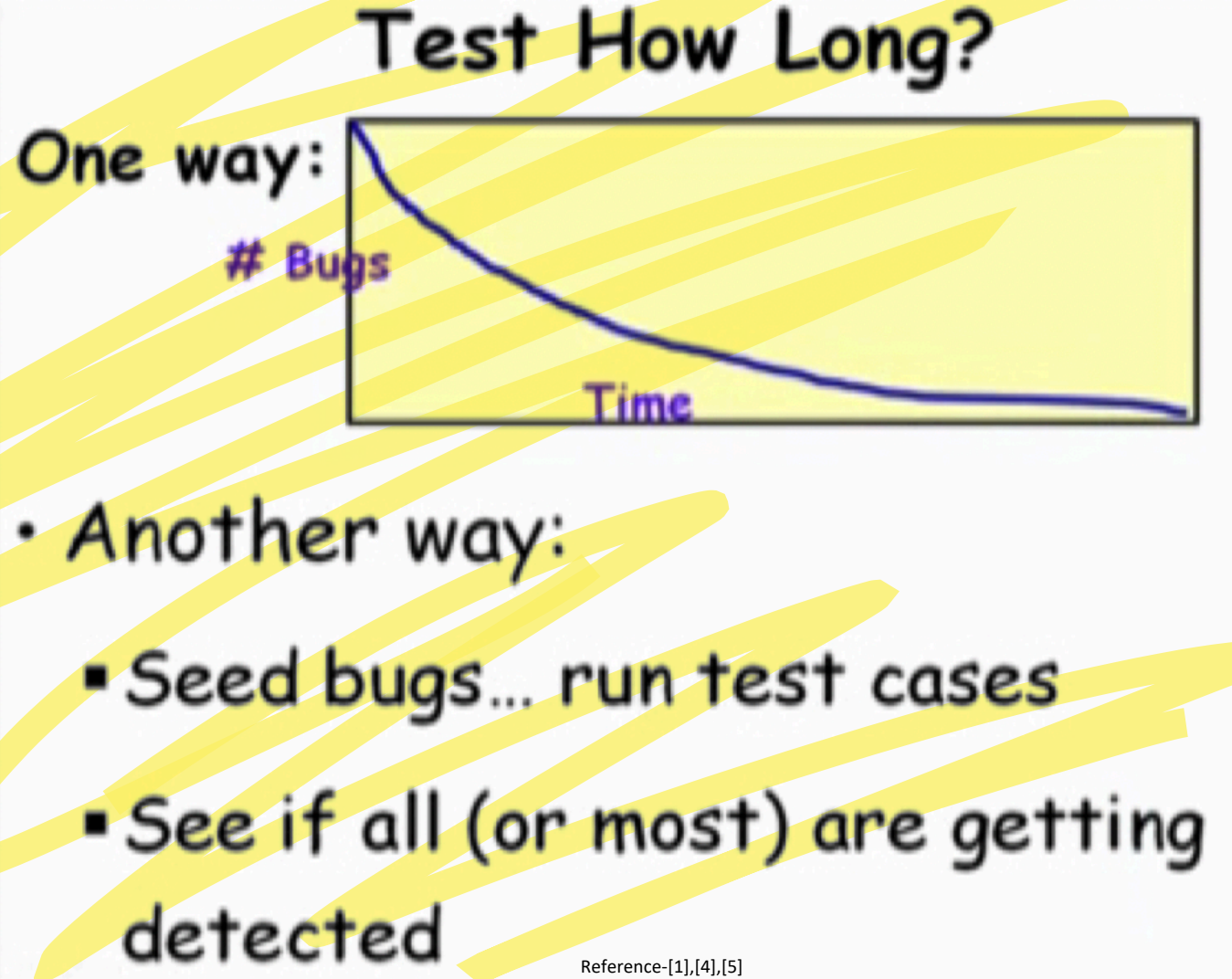
Reference-[1],[4],[5]

What Should We Test?

- We require effective planning, strategies and sufficient resources even to target the minimum possible bottom line.
- We should also check the program for very large numbers, very small numbers, numbers that are close to each other, negative numbers, some extreme cases, characters, special letters, symbols and some strange cases.

Reference-[1],[4],[5]

When to stop testing??

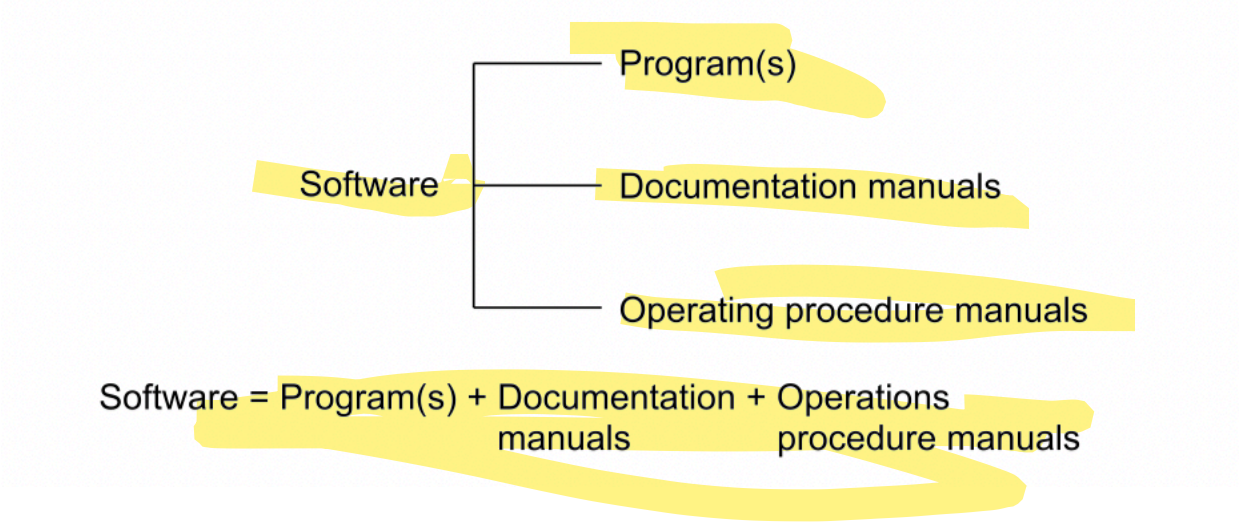


Reference-[1],[4],[5]

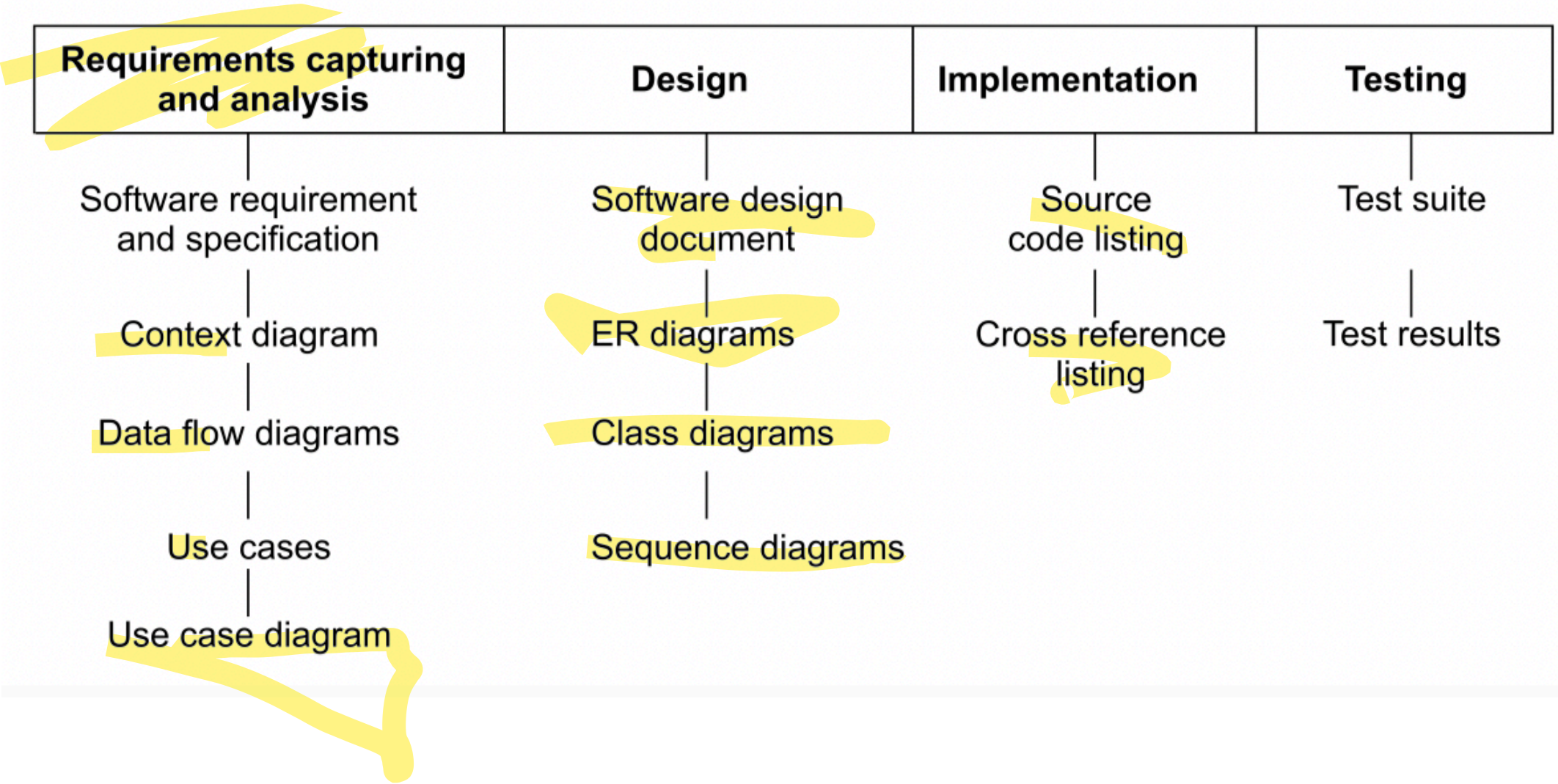
SOME TERMINOLOGIES

Software and Program

- The program is a combination of source code and object code. Every phase of the software development life cycle requires preparation of a few documentation manuals



Reference-[1],[4],[5]



Reference-[1],[4],[5]

Software and Program

- Operating procedure manuals consist of instructions to set up, install, use and to maintain the software.

Reference-[1],[4],[5]

Fault, Error, Bug, Failure

- All terms are used interchangeably although error, mistake and defect are synonyms in software testing terminology.
- When we make an error during coding, we call this a 'bug'. Hence, error / mistake / defect in coding is called a bug.
- A fault is the representation of an error where representation is the mode of expression such as data flow diagrams, ER diagrams, source code, use cases, etc. If fault is in the source code, we call it a bug.

Reference-[1],[4],[5]

Fault, Error, Bug, Failure

- A failure is the result of execution of a fault and is dynamic in nature.
- When the expected output does not match with the observed output, we experience a failure.
- The program has to execute for a failure to occur.
- A fault may lead to many failures.
- A particular fault may cause different failures depending on the inputs to the program.

Reference-[1],[4],[5]

Test, Test Case and Test Suite

- Test and test case terms are synonyms and may be used interchangeably.
- A test case consists of inputs given to the program and its expected outputs. Inputs may also contain precondition(s) (circumstances that hold prior to test case execution), if any, and actual inputs identified by some testing methods.
- Expected output may contain post-condition(s) (circumstances after the execution of a test case), if any, and outputs which may come as a result when selected inputs are given to the software. Every test case will have a unique identification number.

Reference-[1],[4],[5]

Alpha, Beta and Acceptance Testing

- **Acceptance Testing:** This term is used when the software is developed for a specific customer.
- The customer is involved during acceptance testing. He/she may design adhoc test cases or well-planned test cases and execute them to see the correctness of the software.
- This type of testing is called acceptance testing and may be carried out for a few weeks or months. The discovered errors are fixed and modified and then the software is delivered to the customer.

Reference-[1],[4],[5]

Alpha and Beta Testing

- **Alpha and Beta Testing:** These terms are used when the software is developed as a product for anonymous customers. Therefore, acceptance testing is not possible.
- Some potential customers are identified to test the product.
- The alpha tests are conducted at the developer's site by the customer.
- These tests are conducted in a controlled environment and may start when the formal testing process is near completion.

Alpha and Beta Testing

- The beta tests are conducted by potential customers at their sites.
- Unlike alpha testing, the developer is not present here.
- It is carried out in an uncontrolled real life environment by many potential customers.
- Customers are expected to report failures, if any, to the company. These failure reports are studied by the developers and appropriate changes are made in the software.

Testing, Quality Assurance and Quality Control

- The purpose of testing is to find faults and find them in the early phases of software development.
- We remove faults and ensure the correctness of removal and also minimize the effect of change on other parts of the software.

Reference-[1],[4],[5]

Testing, Quality Assurance and Quality Control

- The purpose of QA activity is to enforce standards and techniques to improve the development process and prevent the previous faults from ever occurring. A good QA activity enforces good software engineering practices which help to produce good quality software.
- The QA group monitors and guides throughout the software development life cycle.
- This is a defect prevention technique and concentrates on the process of the software development.
- Examples are reviews, audits, etc.

Reference-[1],[4],[5]

Testing, Quality Assurance and Quality Control

- Quality control attempts to build a software system and test it thoroughly. If failures are experienced, it removes the cause of failures and ensures the correctness of removal.
- It concentrates on specific products rather than processes as in the case of QA.
- This is a defect detection and correction activity which is usually done after the completion of the software development.
- An example is software testing at various levels.

Reference-[1],[4],[5]

Static and Dynamic Testing

- **Static testing refers** to testing activities without executing the source code. All verification activities like inspections, walkthroughs, reviews, etc. come under this category of testing.
- **Dynamic testing** refers to executing the source code and seeing how it performs with specific inputs. All validation activities come in this category where execution of the program is essential.

Reference-[1],[4],[5],web resource

Testing and Debugging

- The purpose of testing is to find faults and find them as early as possible.
- When we find any such fault, the process used to determine the cause of this fault and to remove it is known as debugging.
- These are related activities

Verification and Validation

- Verification: As per IEEE [IEEE01], “It is the process of evaluating the system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.”
- We apply verification activities from the early phases of the software development and check / review the documents generated after the completion of each phase.
- Hence, it is the process of reviewing the requirement document, design document, source code and other related documents of the project.
- This is mainly manual testing and involves only looking at the documents in order to ensure what comes out is what we expected to get.

- Verification is the process of determining:
  - Whether output of one phase of development conforms to its previous phase.

Reference-[1],[4],[5]

Verification and Validation

- Validation: As per IEEE [IEEE01], “It is the process of evaluating a system or component during or at the end of development process to determine whether it satisfies the specified requirements.” It requires the actual execution of the program.
- It is dynamic testing and requires a computer for execution of the program.
- we experience failures and identify the causes of such failures.
- Hence, testing includes both verification and validation. Thus

Testing = Verification + Validation

Reference-[1],[4],[5]

Verification	Validation
Are you building it right?	Have you built the right thing?
Checks whether an artifact conforms to its previous artifact.	Checks the final product against the specification.
Done by developers.	Done by Testers.
Static and dynamic activities: reviews, unit testing.	Dynamic activities: Execute software and check against requirements.

Reference-[1],[4],[5]

LIMITATIONS OF TESTING

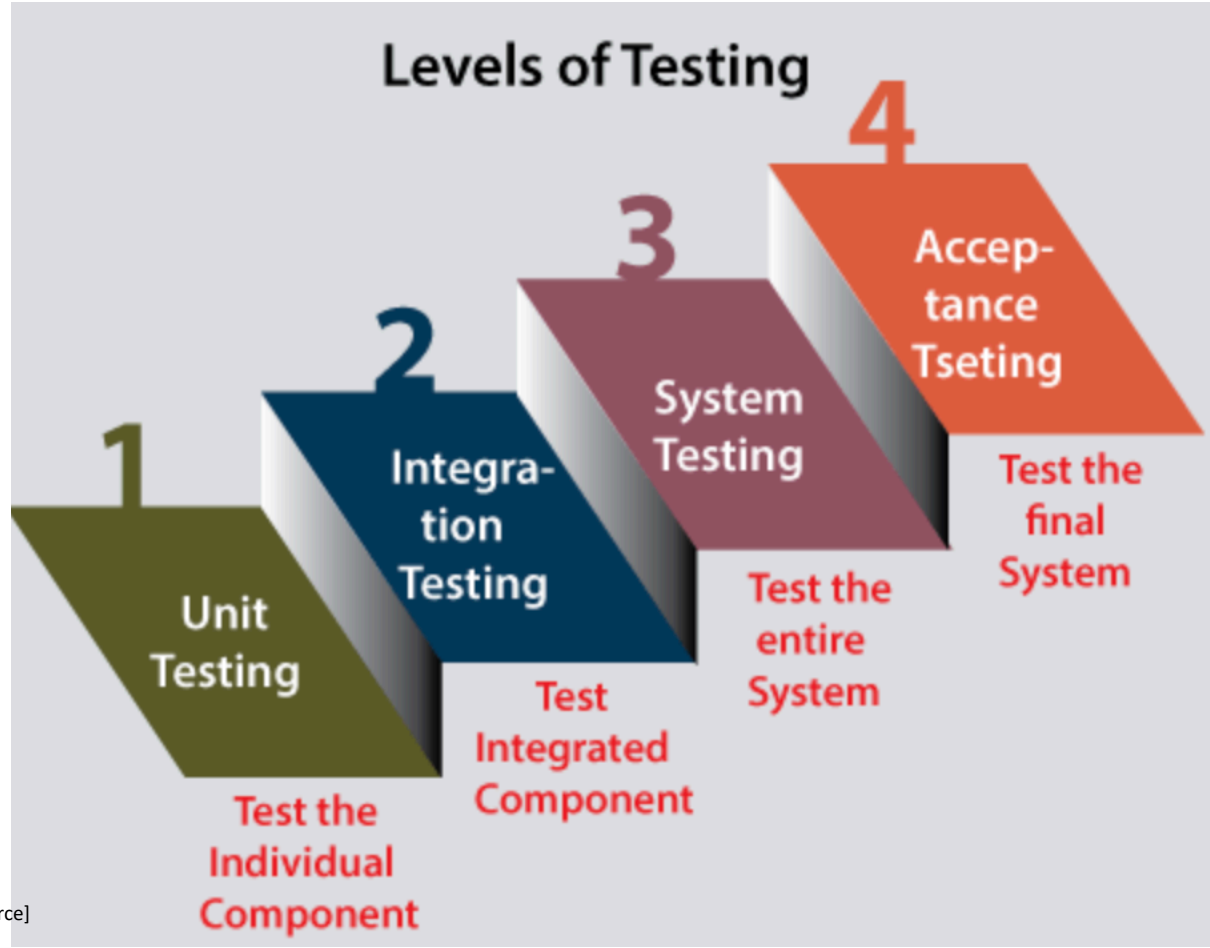
- We want to test everything before giving the software to the customers. This ‘everything’ is very illusive and has many meanings. What do we understand when we say ‘everything’? We may expect one, two or all of the following when we refer to ‘everything’:

- (i) Execute every statement of the program
- (ii) Execute every true and false condition
- (iii) Execute every condition of a decision node
- (iv) Execute every possible path
- (v) Execute the program with all valid inputs
- (vi) Execute the program with all invalid inputs

Reference-[1],[4],[5]

Testing Levels

- UNIT TESTING
- INTEGRATION TESTING
- SYSTEM TESTING
- ACCEPTANCE TESTING
- REGRESSION TESTING



Reference-[1],[4],[5,web resource]

Testing Levels

- Unit testing-As the software getting developed unit needs to be tested as soon as it has been developed.
- Then after all the units have been tested need to do the Integration testing, where we integrate the different units and then test if the integration they are working all right.
- The integration testing is done after the unit testing and we check whether the units are integrating or they are working together well.

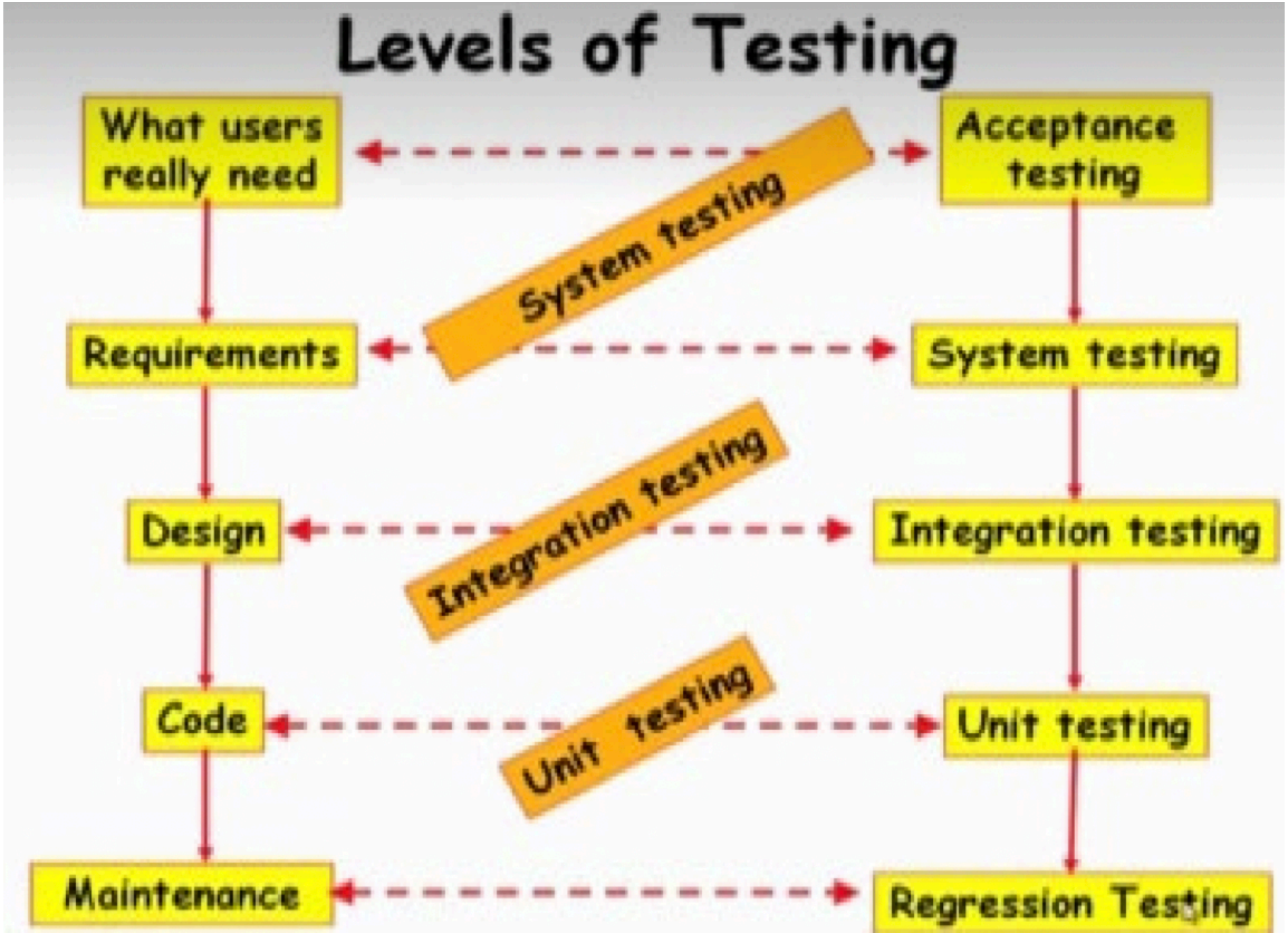
Reference-[1],[4],[5]

Testing Levels

- Then finally, the System testing - where all the units have been integrated and tested against the specification to check if the software is working as you are specified.
- Acceptance testing, which is used to evaluate whether a specification or the requirements are met as per its delivery..
- Regression testing. This is undertaken during maintenance, so after the software is released there can be bug reports or enhancement reports, putting requirement and so on and the software gets changed.
- And whenever there is change, we need to check whether the changed
- part is working all right, not only that we need to check whether the other parts which have not being changed whether they are continuing to work all right

Reference-[1],[4],[5]

- **Unit testing**
  - Test each module (unit, or component) independently
  - **Mostly done by developers of the modules**
- **Integration and system testing**
  - Test the system as a whole
  - **Often done by separate testing or QA tea**
- **Acceptance testing**
  - **Validation of system functions by the customer**


Levels of Testing
What users really need
Acceptance testing
System testing
Requirements
System testing
Integration testing
Design
Integration testing
Unit testing
Code
Unit testing
Maintenance
Regression Testing

LEVELS of testing

- The types of bug during-
- UNIT TESTING
  - can detect an algorithm error or a programming error,
- INTEGRATIN TESTING
  - One example of a bug detected during unit testing is miss match of parameters. In place of i and j are two different variables, int variables instead of passing i and j it is passing j and i. So it has just interchanged the two variables that were expected. That is a kind of integration bug.
- SYSTEM TESTING
  - Functionality bugs and performance bugs
  - unit and integration do not look at performance aspects, usability aspects, so these are detected during system testing.

Reference-[1],[4],[5]

Performance Testing

- The performance test deal with checking the non functional requirement.
- The functional tests, they check the functional requirements in the SRS document, the requirements document. Whereas the performance tests, they check the non-functional requirements as documented in the requirements document.
- There are many types of performance tests .for example; Response times, Throughput, Usability, Stress working under stressed conditions like number of inputs per unit time is large and so on, Recovery, Configuration.

Reference-[1],[4],[5]

Overview of Activities During System and Integration Testing

- Test Suite Design
- Run test cases
- Check results to detect failures.
- Prepare failure list

Tester

- Debug to locate errors
- Correct errors.

Developer

7

Reference-[1],[4],[5]

Pesticide Effect

- Errors that escape a fault detection technique:
  - Can not be detected by further applications of that technique

Reference-[1],[4],[5]

Pesticide Effect

- Bugs that escape detection by some fault detection technique cannot be detected by further application of that technique.
- In testing we have many testing techniques we can call them as Filters.
- Initially there are some bugs and we apply one type of filter.
- The bugs that are survive a filter cannot be eliminated by further application of that filter and also new bugs get introduced by correcting the bugs that where adjusting and also newer development and so on. So, new types of bugs are there.
- As the system development proceeds, we use these filters, and we need to use many types of filters to effectively reduce the bugs. So many test methodologies have to be used, dozens of them.

Reference-[1],[4],[5]

V SHAPED SOFTWARE LIFE CYCLE MODEL

- The V shaped model is the modified form of the waterfall model with a special focus on testing activities.
- The waterfall model allows us to start testing activities after the completion of the implementation phase.
- This was popular when testing was primarily validation oriented. Now, there is a shift in testing activities from validation to verification where we want to review / inspect every activity of the software development life cycle.

Reference-[1],[4],[5]

V SHAPED SOFTWARE LIFE CYCLE MODEL

- We want to involve the testing persons from the requirement analysis and specification phase itself.
- They will review the SRS document and identify the weak areas, critical areas, ambiguous areas and misrepresented areas.
- This will improve the quality of the SRS document and may further minimize the errors.

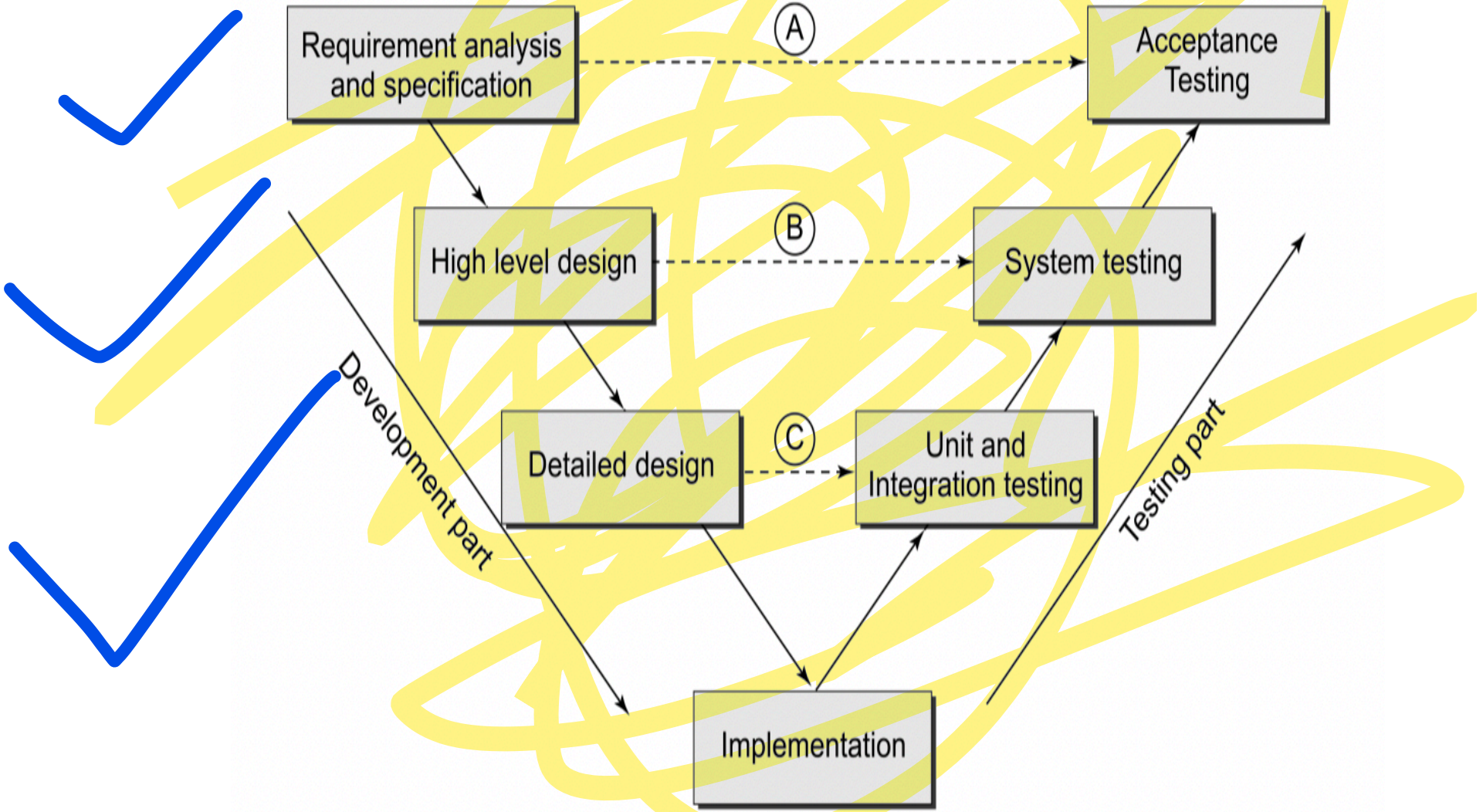
Reference-[1],[4],[5]

V SHAPED SOFTWARE LIFE CYCLE MODEL

- These verification activities are treated as error preventive exercises and are applied at
- requirements analysis and specification phase,
- high level design phase, detailed design phase
- and implementation phase.

We not only want to improve the quality of the end products at all phases by reviews, inspections and walkthroughs,

Reference-[1],[4],[5]



(A) Acceptance test case design and planning

(B) System test case design and planning

(C) Unit and integration test case design and planning

Reference-[1],[4],[5]

V SHAPED SOFTWARE LIFE CYCLE MODEL

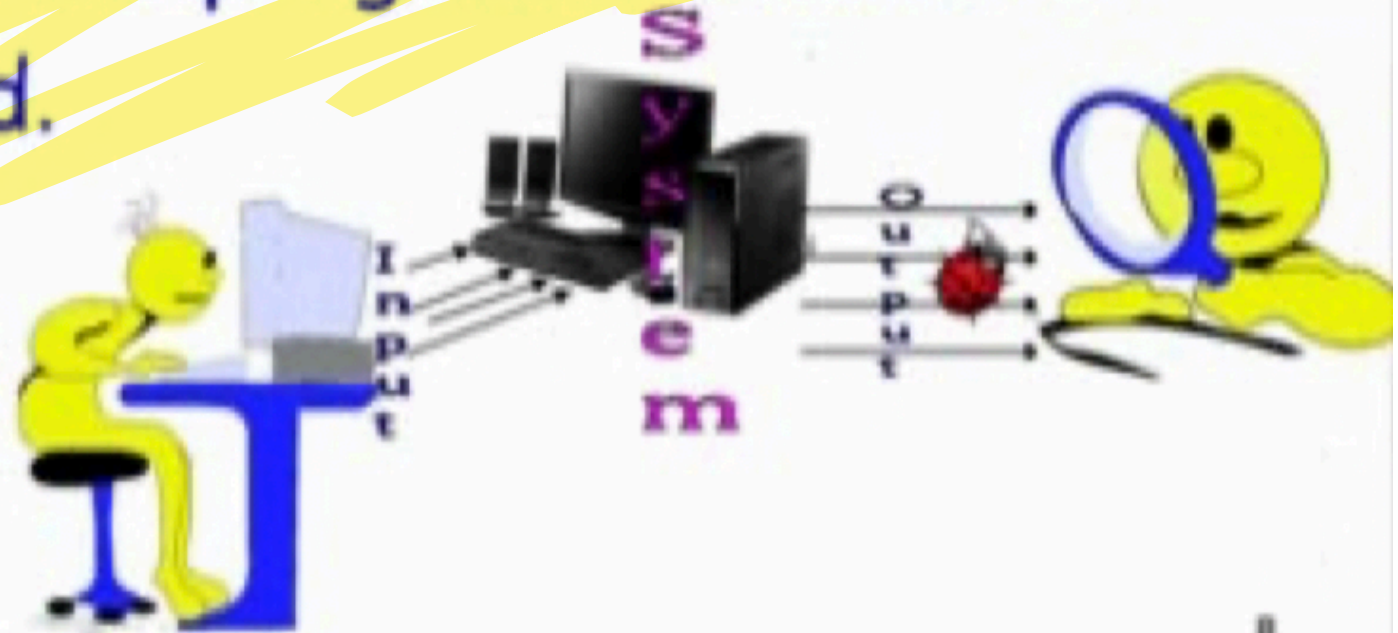
- The designing of test cases-
- after requirement analysis and specification phase,
- high level design phase, detailed design phase
- and implementation phase

may help us to improve the quality of the final product and also reduce the cost and development time.

Reference-[1],[4],[5]

How to Test?

- Input test data to the program.
- Observe the output:
  - Check if the program behaved as expected.



Reference-[1],[4],[5]

Examine Test Result

- If the program not behave as expected
  - Note the condition under which it is failed
  - Later Debug and correct

Reference-[1],[4],[5]

Testing Facts

- Testing is getting more complex every year
  - (because, the programs themselves are becoming large and complex. New programming paradigms have come up for example, web based software or may be embedded software and software running on mobile phones and so on. And also lot of tools, new testing techniques)
- Consume largest time among all the development activities
  - Large man power

  (All companies spend at least 50 percent effort on testing and 50 percent effort on specifying, designing, coding and so on.)

Reference-[1],[4],[5]

Testing Facts

- Testing is not viewed as challenging
  - (but now the testers need to have good knowledge of different testing techniques and there are large number of them, and also the automated tools that have become available must develop a proficiency with those)

Reference-[1],[4],[5]

References

1.Software Testing by Yogesh singh

2.Paul C. Jorgensen, “Software Testing: A Craftsman's Approach”, Auerbach Publications,4th Edition, 2013/ Latest Edition.

3. Aditya P. Mathur, “Foundations of Software Testing”, Pearson, 2nd Edition, 2013/ Latest Edition.

4. https://nptel.ac.in/courses/106/101/106101163/

5. https://nptel.ac.in/courses/106/105/106105150/