Course for Software Testing

BS (Computer Science)
Spring Semester
February 2014
Foundation University Rawalpindi Campus

Instructor: Sohaib Altaf

sohaib@hybriditservices.com

http://www.hybriditservices.com/course/FU-BSSE-ST

Course Introduction

Course Name: Software Testing

Lecture Time:

Instructor: Sohaib Altaf

Course Cr. Hrs: **(Theory)** 03 Hours+(Lab) 03 Hours

Class Website: Course Materials: Syllabus, Lecture Notes, Projects & Assignments, Primary & Reference Text Information, Related Material and Assignments will be available on www.hybriditservices/course/FU-BSSE-ST

Contact Details: e-mail: sohaib@hybriditservices.com

Cell: + 92 (333) 5107494

Off: + 92 (051) 5596168

Course Objective

- To study fundamental concepts in software testing, including software testing objectives, process, criteria, strategies, and methods.
- To discuss various software testing issues and solutions in software unit test; integration, regression, and system testing.
- To learn how to planning a test project, design test cases and data, conduct testing operations, manage software problems and defects, generate a testing report.
- To understand software test automation tools
- To learn how to write software testing documents, and communicate with engineers in various forms.
- To gain the techniques and skills on how to use modern software testing tools to support software testing projects.
- To expose the advanced software testing topics, such as object-oriented software testing methods, and component-based software testing issues, challenges, and solutions.

Rules to be Obeyed

- **Quizzes are always unannounced.**
  - They include all topics covered in previous classes.
  - So come prepared in each class.

- **Any question can be asked at any time in any class**
  - One person at one time
  - Focus on class discussion, it will be evaluated.
  - Logical questions which initiate thinking, are encouraged
  - Be relevant

- Comments are discouraged and questions of any level are always encouraged at all time in and out side the class
- No Discussion or Whispering during the Lecture
- Mobile phones must not be used and must always be on Vibrator or Switched off

Course Grade

- You all have 100% marks at start of the course
- During the course you will be tested by quizzes, assignments, mid-term, project, final papers & others form of evaluations.
- Lets see how can you retain your marks till the end of the course.

Text

Primary Text

- Aditya P. Mathur. *Foundations of Software Testing*. Pearson Education, 2008.

Reference Text

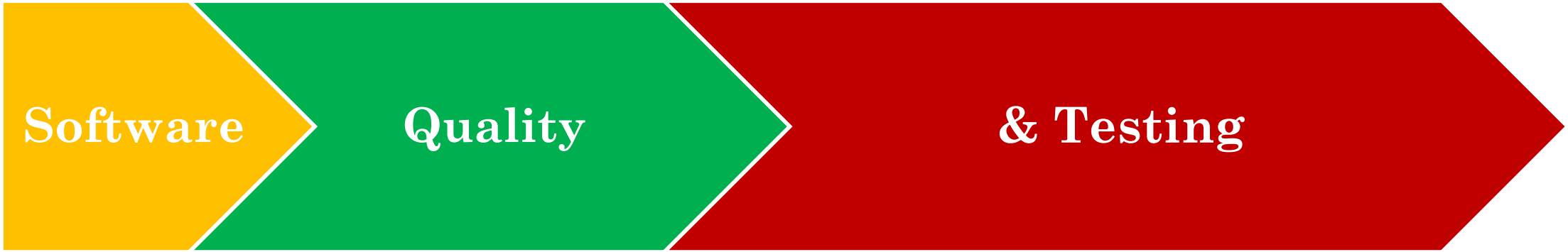
- **Software Quality Engineering:** Testing, Quality Assurance, and Quantifiable Improvement, 2009 by Jeff Tian **(find e-copy at website)**
- Software Testing & and Internationalization by Manfred Rätzmann & Clinton De Young **(find e-copy at website)**
- Software Engineering, A Practitioner's Approach by Roger S. Pressman

Student Assignments

- **Groups Work**
- **Project Artifacts**
- **Presentation on Selected Topics**
- **Assignments: Case Study & Any other Reading Material, Industrial Examples**

LECTURE 1

The beginning…


Software
Quality
& Testing

What is Software?

- Software can be considered a product of engineering just like an airplane, automobile, television, or an other object that requires a high degree of skill to turn a raw material into a usable product.

- But software is developed or engineered, not manufactured! No wear-out

- Software is a set of programs or data combined with its documentation which is helpful or even needed to run the application. (Hesse et al. 1984)

What is Software?

Software can be of different types (Different Classification):

- System software
- Engineering / Scientific software
- Product-line software
- OO Software
- Telecommunication software
- Application software
- Embedded software
- Web-applications
- Medical software
- Dependable software i.e.
- Airplane Software

Different Taxonomies?

Each Taxonomies is exhibiting different characteristics e.g.

- Safety is more critical for Dependable Software
- Accuracy is more critical for Medical Software
- Look & Feel is more critical for Web-applications

What is Software?

Taxonomies: (common classes exhibit common properties)

- System software: Drivers, operating system components, compilers,
- Application software: Stand-alone software for specific business need, transaction processing system
- Scientific software: Stress analysis software, simulation software
- Embedded software: Reside within a product or system, keypad control for microwave,
- Product-line software: Word-processing, spread sheets
- Artificial Intelligence software: Game playing, expert systems

Other Taxonomies…

- Web Application (or Desktop Application)
- Static or Dynamic, Hard & Non-Hard Problems (A.Davis)

What is Software System?

- System Context
- Set of elements having specified relationships among each other (Duden, 1974)
- Software as Part of the System
- To understand software and to define its requirements, it is necessary
  - to consider software only as a small part of a system (in case of software-intensive / complex system).
  - Need to understand all its interfaces
- **Examples:**
- Information system - Organization context; all subsystem, organizational policies i.e. patient information system, other sub-system of hospital
- Embedded system - Physical context, environment i.e. multimedia projector

What is Engineering?

The profession of:

- creating cost-effective solutions -- > economy solutions
- to practical problems: whose solution matter to people outside the engineering domain
- by applying scientific knowledge: solves by *science*, *mathematics*, *design analysis*, *technology*, *tools*, *methods*, *techniques & established practices*
- In service of mankind: besides for customers, to develop products for use by society, safety, security, ethical
- *... and taking responsibility for them!*

WHAT IS ENGINEERING?

- Scientific Principle
- Cost Effective
- Practical Problems
- Useful Solutions
- Service to Mankind

1

Tools (Quality?)

2

Methods, Procedures,
Production, Principles
People, Training (Quality?)

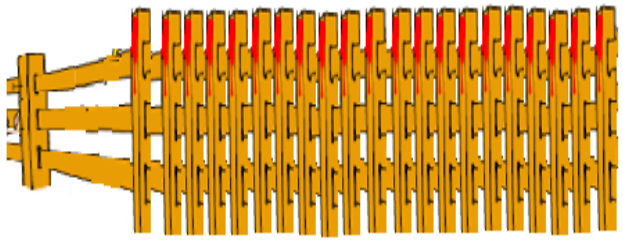
- - - - - - - - - -

4

Support Service
(Quality?)

Dispatch
(Quality?)

3



End Product (Quality?)

5
Quality of
Service or
Product ?

WHAT IS ENGINEERING?

Production Guide

Methods, Procedural Guide

Dispatch Guide

Training Guide

Support Guide

Warranty Guide

Inspection Guide

Quality Guide

Testing Guide

Tools Guide

Engineering Guides….

What is Software Engineering?

The profession of:

- creating cost-effective software solutions -- > economy software solutions
- to practical problems: whose solution matter to people outside the engineering domain
- by applying scientific knowledge: solves by *science*, *mathematics*, *design analysis*, *technology*, *tools*, *methods*, *techniques & established practices*
- In service of mankind: besides for customers, to develop products for use by society, safety, security, ethical
- *... and taking responsibility for them!*

SOFTWARE ENGINEERING AS ENGINEERING DISCIPLINE

Scientific principle…! - >

Software Engineering is more than just Programming:

This process generally starts long before a Line of Code (LOC) is written & continue long after the initial version of program has been completed

Software Engineering is the application of scientific principles to

1) the orderly transformation of a problem into a working software solution and
2) the subsequent maintenance of the software until the end of its useful life

SOFTWARE ENGINEERING?

SE Tools (Quality?)

1

2

SE Techniques, Methods, Procedures, Methodologies, Principles, Production, People, Training (Quality?)

4

Support (Quality?)

3

SE Deployment (Quality?)

SE End Product (Quality?)

5

Quality (Software Product) = Quality (Software Process)

SOFTWARE ENGINEERING?

SE Development Guide

SE Methods, Procedural Guide

SE Deployment Guide

SE Training Guide

SE Support Guide

SE Warranty Guide

SE Inspection Guide

SE Quality Guide

SE Testing Guide

SE Tools Guide

Ongoing research on more…

Software Engineering Guides….

Major Software Failures of 2011

Deficiencies in software quality often results

- in costly emergency fixes
- damage to a brand's reputation, software company defaulters – large repayments

Software defects are extremely costly in term of

- money
- loss of life
- reputation

Major Failures

Cash machine bugs, benefits the customers by giving them extra money (Financial / Money Loss)

> *An Australian bank **began giving out large sums of money** from 40 Cash Machines cross one city. Officials at the company said they were **operating in stand-by mode, so could not identify the account balances** of customers.*

MAJOR SOFTWARE FAILURES OF 2011

Major Failures

-22 people wrongly arrested in Australia due to failures in new NZ $54.5 million courts computer system (Human Loss)

*A new NZ $54.5 million ($42.7 million or £26.8 million) computer system linking New South Wales courts and allowing documents to be lodged electronically led to **damages claims for unlawful arrest** and malicious prosecution, after 3600 defects in the electric transfers of data from the courts to the police's database led to the **wrongful arrest of 22 individuals.***

-**Computer system bugs cause Asian banking facilities' downtime (Customer Loss, R**eputation damages**)**

*Computer system problems at one of Japan's largest banks resulted in a nationwide ATM network of more than **5,600 machines going offline for 24 hours, i**nternet banking services being shut down for three days, delays in salary payments worth $1.5 billion (£939 million) into the accounts of 620,000 people and a **backlog of more than 1 million unprocessed payments worth around $ 9 billion (£5.64 billion).***

Some Failure Statistics: Economy

- In the United States, we spend more than **$250 billion** each year on IT application development of approximately **175,000 projects.**

- The average cost of a development project for a
  - **large company is $2,322,000;**
  - **for a medium company, it is $1,331,000;**
  - **and for a small company, it is $434,000.**

- **The Increasing Trend***
  - In **2000**, there were **300,000 new IT projects**
  - In **2001**, over **500,000 new IT projects were started**

***The Standish Group, "CHAOS 2001: A Recipe for Success"**

CHAOS REPORT (2004) SOFTWARE
PROJECTS HAVE A TERRIBLE TRACK RECORD

RESOLUTION OF PROJECTS

This year's results show that 29% of all projects succeeded (delivered on time, on budget, with required features and functions); 53% are challenged (late, over budget and/or with less than the required features and functions); and 18% have failed (cancelled prior to completion or delivered and never used), as shown in Figure 2.0.

Improvement from 1994 – 2004, but yet new challenges ! i.e. Geographically Distributed Teams, Outsourcing…

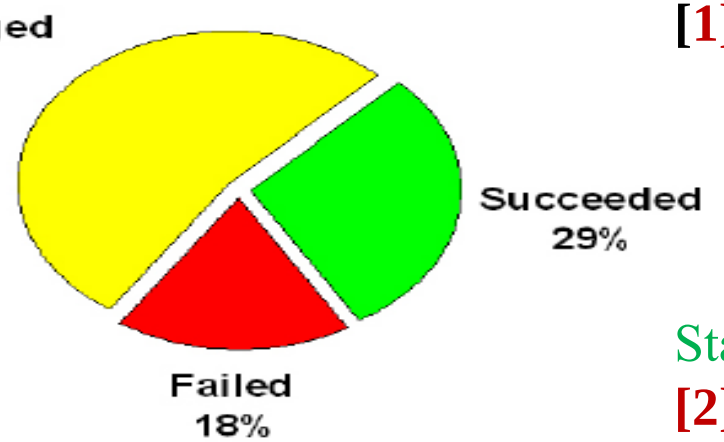


Figure 2.0

In Challenged Projects
1994 : 189 % of increase from their original Estimate

[1] Standish Group Inc., 2004
Success…….. 29 %
Challenged…… 53 %
Failed…….. 19 %

Statistics…before Ten Years
[2] Standish Report in 1994
Success…….. 16.2 %
Challenged…… 52.7 %
Failed…….. 31.1 %

(Quality Development process) relation to defects

Majority of defects are Introduced in earlier phases

Phase	Percentage of defects	Effort to fix defects
Requirements	56	82
Design	27	13
Code	7	1
Others	10	4

Relative cost of fixing defects at… (no. of times, cost ratio %)

1:15 – 1:40 times more cost if Defects accumulates at unit testing stage

1:30 – 1:70 times more cost if defects accumulates at acceptance testing Stage

Phase in which found	Cost Ratio
Requirements	1
Design	3 – 6
Coding	10
Unit/Integration testing	15 – 40
System/Acceptance testing	30 – 70
Production	40 – 1000

Some Causes of software defects

1. Faulty requirements definition
2. Client-developer communication failures
3. Deliberate deviations from software requirements
4. Logical design errors
5. Coding errors
6. Non-compliance with documentation and coding instructions
7. Shortcomings of the testing process
8. User interface and procedure errors
9. Documentation errors
10. Others…

Errors

- Errors are a part of our daily life.
- Humans make errors in their thoughts, actions, and in the products that might result from their actions.
- Errors occur wherever humans are involved in taking actions and making decisions.

These fundamental facts of human existence make testing an essential activity.

Errors: Examples

Area	Error
Hearing	Spoken: He has a garage for repairing *foreign* cars. Heard: He has a garage for repairing *falling* cars.
Medicine	Incorrect antibiotic prescribed.
Music performance	Incorrect note played.
Software	Operator used: $\neq$, correct operator: $>$. Identifier used: new_line, correct identifier: next_line. Data conversion from 64-bit floating point to 16-bit integer not protected (resulting in a software exception).
Speech	Spoken: *waple malnut*, intent: *maple walnut*. Spoken: *We need a new refrigerator*, intent: *We need a new washing machine*.
Sports	Incorrect call by the referee in a tennis match.

Errors – Example

- An instructor administers a test to determine how well the students have understood what the instructor wanted to convey
- A software developer tests the program developed to determine if it behaves as desired.

In each of above two cases, there is an attempt by a tester to determine if the human thoughts, actions, and products behave as desires Errors

Errors – Example

- Deviation from expected may not be due to an error for one or more reasons. Suppose the tester wants to test a program to sort a sequences of integer. The program can sort an input sequence on both descending or ascending orders depending on the request made.
- Now suppose that the tester wants to check if the program sorts an input sequence in descending order. To do so, he types in an input sequence and a request to sort the sequence in descending order
    - Suppose the program is correct and produces an output which is the input sequence:
        - A - in descending order
        - B - in ascending order

What is requirement error?
What is programmer error?
What is tester error?
If he hypothesize that sorting is program is incorrect or correct
IF "A" or IF "B"

Quality as Dealing with defects

- When people associate Quality or High Quality with software system, it is an indication that few, if any, defects are expected to occur during its operations or when problems do occur, **the negative impact is expected to be minimized?**.
- Key of the correctness aspect of software quality is the concept of DEFECT: failure, fault and error.
- The term **'DEFECT'** generally refer to the some problem with software either with its external or internal characteristics
- More handling of defects in early stages, less accumulation of defects in later stages i.e. testing stage etc.
- Avoid accumulating defects for testing stage i.e. less will be the confidence on quality

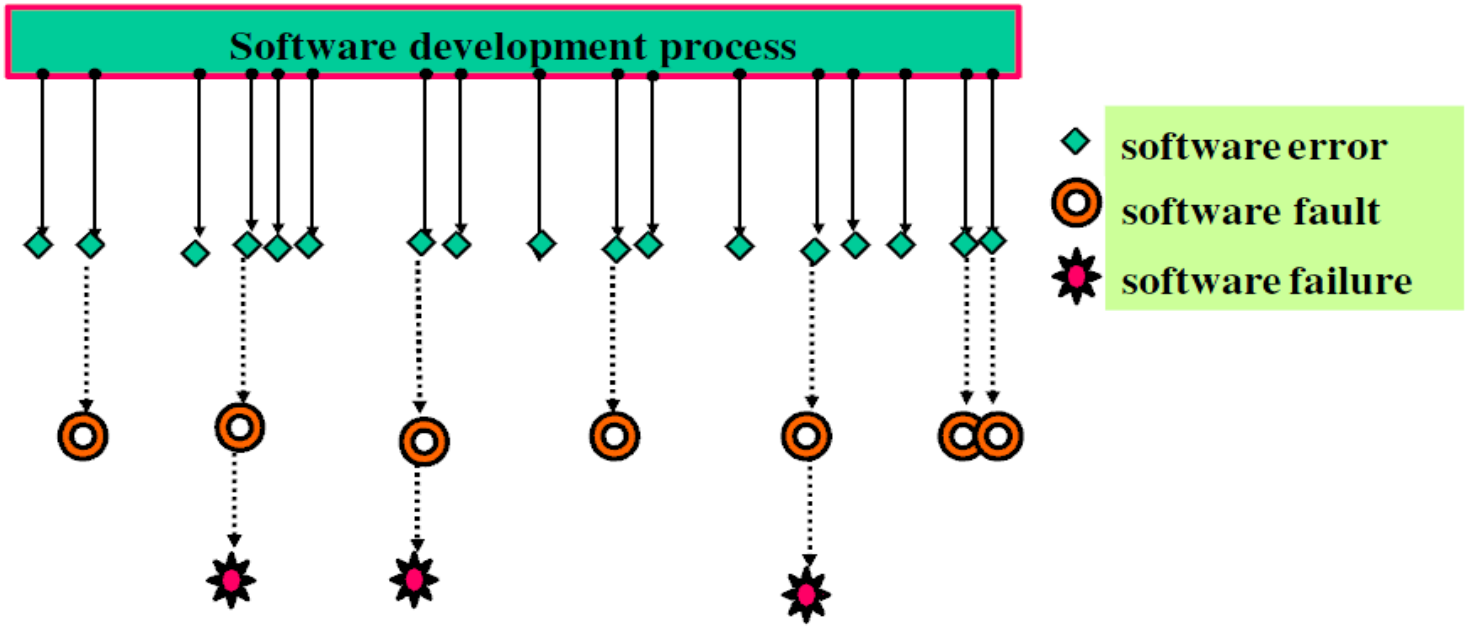
Software Quality as Dealing with defects

- **External & Internal characteristics**
- Dealing with Defects in early phase or later phase? i.e. testing phase
- External & Internal

External (Rely on execution)	**Internal (Don't rely on execution)**
correctness	maintainability
reliability	flexibility & reusability
usability & performance	portability
completeness, consistency, usability	availability of correct & complete documents
robustness	structured & readability

Software Errors, Faults, & Software Failures

- **Software** can be seen as a series of **translation processes**. Each of these translations **produces a work product or deliverable**.
- **Software defects are introduced when there is a failure to completely and accurately translate one representation to another, or to fully match the solution to the problem**.

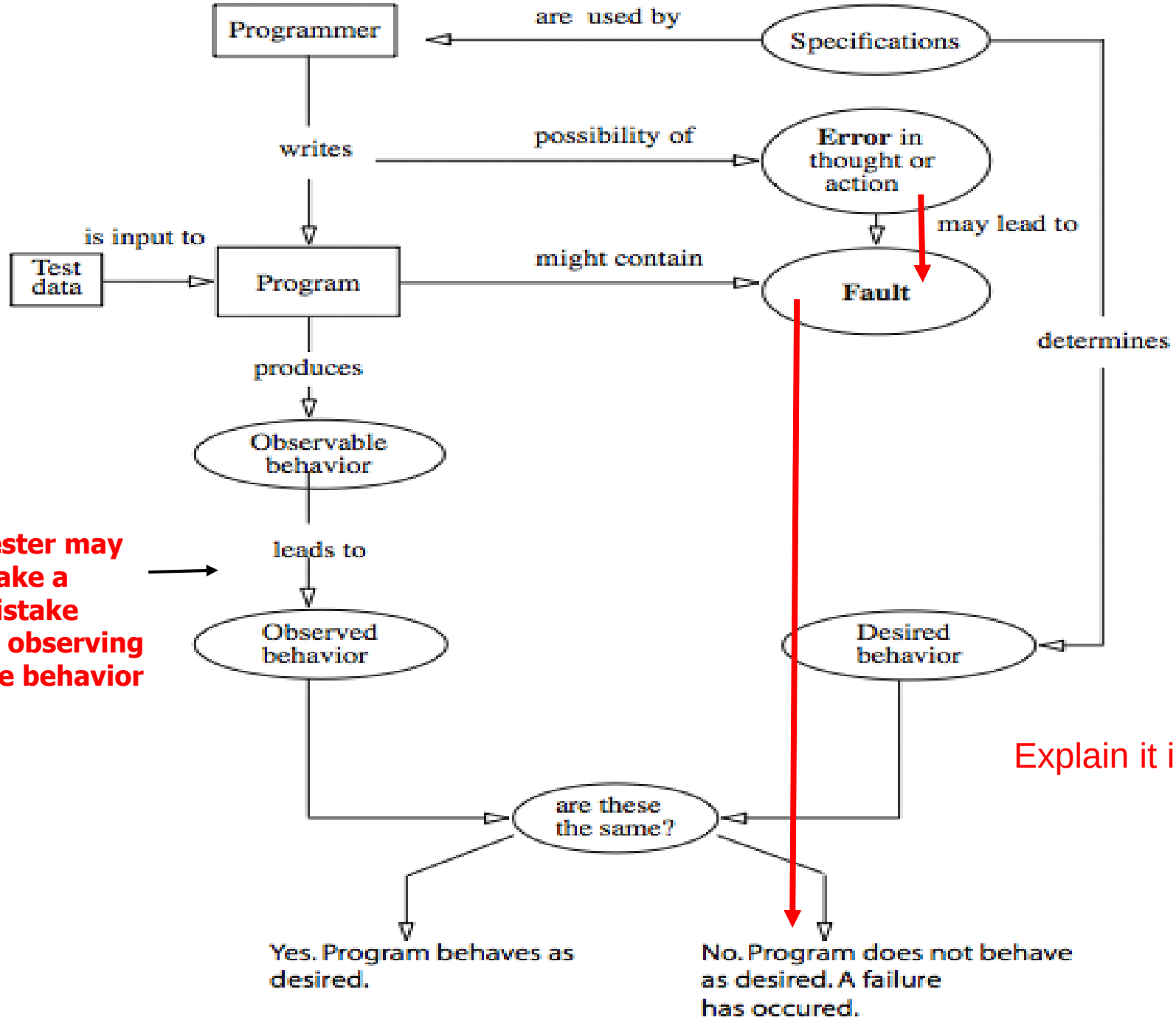


Software Errors, Faults, & Software Failures

- **Error**: A human action that produces an incorrect result.
  - e.g. if is the developer mistake that produce a fault?, if it is caused by human activities such as the typing errors?
- **Fault**: An incorrect step, process, or data definition (created) in a computer program. A failure may be caused by the presence of one or more faults on a given system. However, the presence of a fault in a system may or may not lead to a failure, e.g., a system may contain a fault in its code but on a fragment of code that is never exercised so this kind of fault do not lead to a software failure.

- **Failure**: The inability of a system or component to perform its required functions. It is an observable incorrect behavior or state of a given system. In this case, the system displays a behavior that is contrary to its specifications/requirements. Thus, a failure is tied (only) to system executions/behaviors and it occurs at runtime when some part of the system enters an unexpected state.

Software Error, faults, failures



Tester may make a mistake In observing the behavior

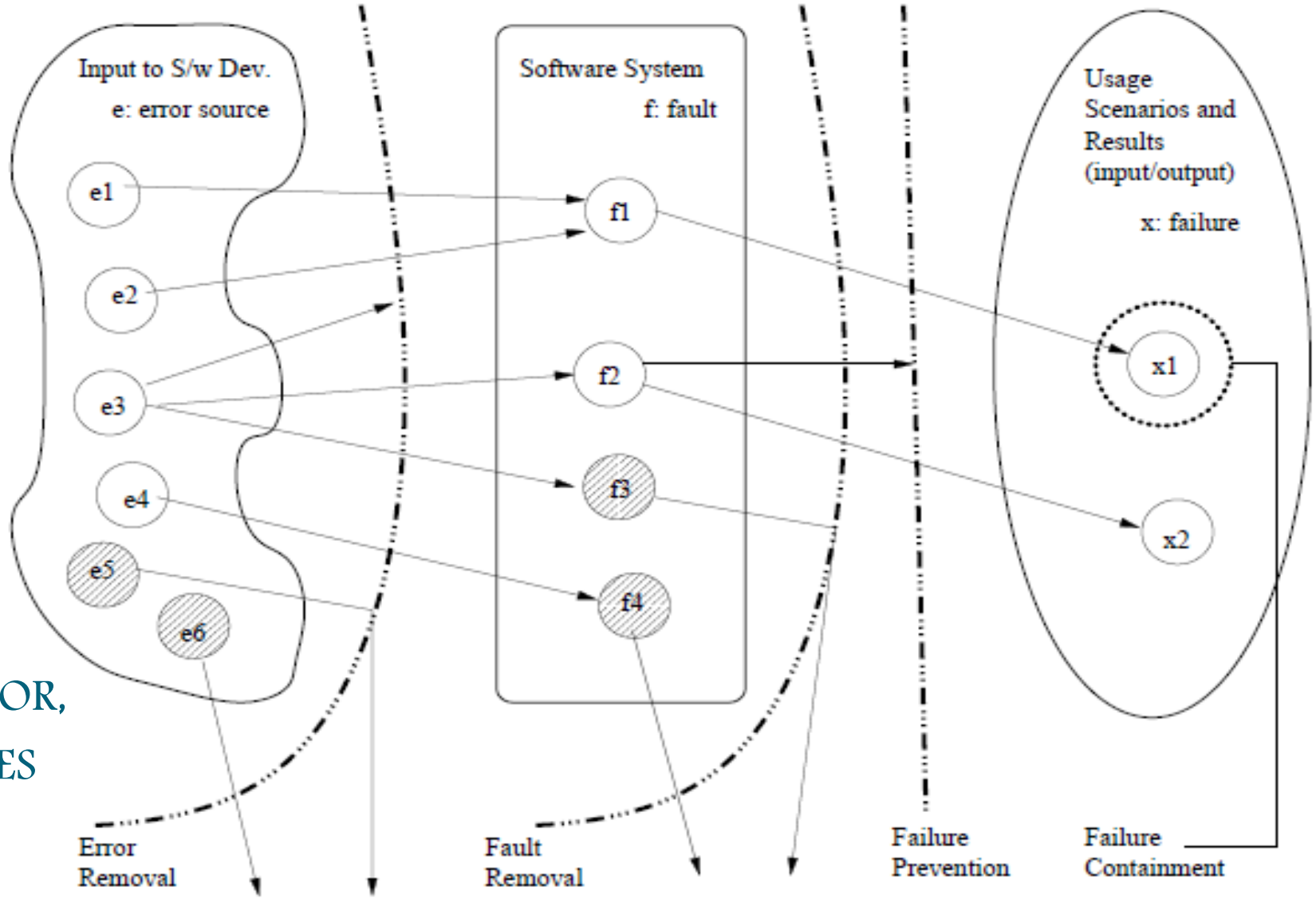
Explain it in your Lab Exercise !

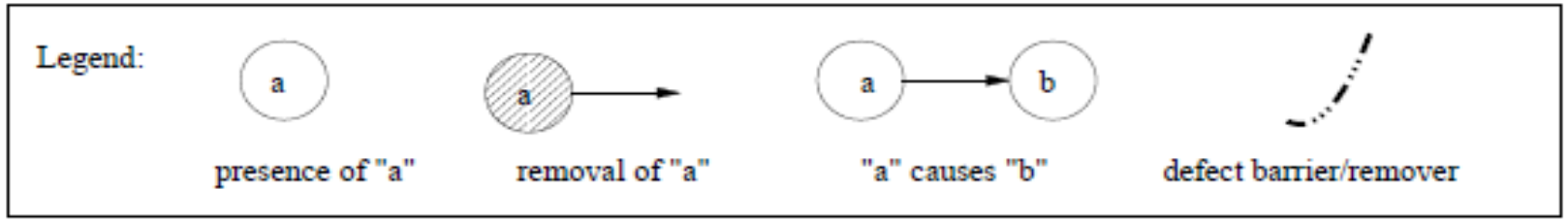
Software Errors, Faults, & Software Failures

errors → faults → failures

- **Errors may cause faults to be injected into the software, and faults may cause failures when the software is executed**
- **A single error may cause many faults, such as in the case that a wrong algorithm is applied in multiple modules and causes multiple faults, and a single fault may cause many failures in repeated executions**
- Observe it in your Lab Exercise

Software Error, faults, failures





Software Errors, Faults, & Software Failures

- Probability of introducing errors, creating faults, causes failures
- Faults which do not cause a failure under the given scenarios are typically called ***dormant*** or ***latent*** faults, which may still cause problems under a different set of scenarios or circumstances. Observe it in your Lab Exercise.
- Failure Containment: Less Chance of Occurrence of failure, hide faults, avoid loss, minimize loss, intimation by alarming i.e. types of software

Software Errors, Faults, & Software Failures

Example …

Lines	Code
1	program double ();
2	var x,y: integer;
3	begin
4	read(x);
5	y := x * x;
6	write(y)
7	end

Program for Adding two Nos.

Observing the Code (Observe the Internal Structure of Program)
Observing the Code (External View: Give input & Observe Output)

Failure: x = 3 means y =9 → IF Failure ?

- This is a failure of the system since the correct output would be *6*

Fault: The fault that causes the failure is in line 5. The * operator is used instead of +.

Error: The error that is reason for this fault injection, may be:

- a typing error (the developer has written * instead of +)
- a conceptual error (e.g., the developer doesn't know how *to double a number*)

Software Errors, Faults, & Software Failures

Example ….

Lines	Code
1	program double ();
2	var x,y: integer;
3	begin
4	read(x);
5	y := x * x;
6	write(y)
7	end

Program for Adding two Nos.

Simple Test for Observing the Failure (External View: Give input & Observe Output):

Correct: if we enter x = 3; it must give y =6
if we enter x = 4; it must give y =8

Failure: if we enter x = 4; and it gives y =16

This is a failure of the system since the correct output must be *8, Therefore our test is* successful

Now how to diagnose the failure….
How fault is produces

Software Errors, Faults, & Software Failures

Example …

Lines	Code
1	program double ();
2	var x,y,z: integer;
3	Begin
4	read(x);
5 6	y := x * x; z := x / y
7 8	write(y) write(z)
8	end

Program for Adding two Nos.

Simple Test for Observing the Fault (Observe the Internal Structure of Program)

Fault: Observe the program; Lines of Code

What to observes? Statements? Syntax?

Line 1, or Line 2 or Line 3, or Line 4 or Line 5 or Line 6, or Line 7

Line 1: Syntax: termination **“ ; ”,** or Line 2: integer definition

Line 5: Statement is y=x * x, Failure? Fault? Fault Lead to Failure

Line 6: Statement is z=x / y, Failure? Fault? Fault Doesn’t Lead to Failure

Line 8: write (z), is it needed?

Run Time Error?

Faults?

How to Fix the Fault? What is need to fix?

Software Errors, Faults, & Software Failures

Example …

Lines	Code
1	program double ();
2	var x,y,z: integer;
3	Begin
4	read(x);
5 6	y := x * x; z := x / y
7 8	write(y) write(z)
8	end

Program for Adding two Nos.

Simple Test for Observing the Fault (Observe the Internal Structure of Program)

What is Error:
Observing the fault, we will diagnose the error,
Error is the source of injection of fault
If it is typing mistake?
Or ignoring the review?
Or knowledge barrier?
Now How to fix error?
If it is Testing or Quality Chapter

Lab. Exercise (Read the Lab Lecture & Example)

Write any Web Type or Desktop Type of Small Program that is doing any simple function with minimum 15 LOC, having two Classes, 2-3 methods, minimum 1 control statement (Loop), & declaration of both local and global variables etc. Generate defects in all three forms; failure, faults, error.

A. ***Review the Lab Lecture & read the example & Submit below exercise in two labs sessions (within lab timings)***
B. ***You can perform this exercise in group of Two Students. In this exercise you will provide two observations as directed.***

1. Choose any problem of your choice that must fulfill the above criteria.
2. Write the requirement statement.
3. Write the Program on paper in simple naturel language, (generate any simple defect here) and record your observation, Print it & Save it in defined folder by your name and reg. no. (for folders consult Lab Admin)
4. In second phase, translate it in Java Language (use any IDE), generate any simple defect here
5. Now Observe the Program (External View: Give input & record Output)
6. Now Observe the Code (Observe the Internal Structure of Program)
7. Observe the Defect in following form; Failure, Fault & Error
8. Give Comments in your program about your observation
9. Make program error free, & save the file in defined folder
10. Type the report in MS Word and submit
11. Lab. Lecture and example is a good reference for you.

Note: You have do this perform this exercise in two lab class. In first lab class (today), follow above steps 1-3. make a paper work & submit and then in second lab (next week) write your program & Submit as directed

WHAT IS QUALITY ?

- **Quality**

- Quality is *conformance* to r**equirements** and **prevention of defects**.

- The degree to which a system, component, or process *meets* **specified requirements.**

- The degree to which a system, component, or process *meets* **customer or user needs or expectations.**

Different Views of Software Quality

- Perceived in different domains differently
- Views according to *(Kitchenham and Pfleeger1996, Pfleeger et al., 2002, software quality:) are:*

User's	fitness for purpose
Manufacturing	conformance to specification
Product	inherent characteristics of the product
Value based	the amount a customer is willing to pay

WHAT IS QUALITY ?

Quality popular view:

- Something "good" but not quantifiable
- Something luxury and classy

Quality professional view:

- Conformance to requirement

  (Crosby, 1979)

  - The requirements are clearly stated and the product must conform to it
  - Any deviation from the requirements is regarded as a defect
  - A good quality product **contains fewer defects**

- Fitness for use (Juran, 1970):

  - Fit to user expectations: meet user's needs
  - A good quality product **provides better user satisfaction**

Different Views of Software Quality

- **User View**
  - The basic building block on which fitness for use is built is the Quality Characteristic.
  - Any feature (property, attribute, etc.) of the products, materials, or processes which is needed to achieve fitness for use is a Quality Characteristic.
- **Manufacturing View**
  - The degree to which a system, component, or process meets
    - Specified requirements, customer/user needs or expectations
    - Conformance to process standards

Different Views of Software Quality

Product View

- In the ***product*** view, the focus is on inherent characteristics in the product itself in the hope that controlling these internal quality indicators will result in improved external product behavior (quality in use)

Value Based View

- In the ***value-based*** view, quality is the customers' willingness to pay for a software.

Software Quality Attributes

- **External & Internal characteristics**
- Dealing with Defects in early phase or later phase? i.e. testing phase
- External (Dynamic) & Internal (Static)

External (Rely on execution)	**Internal (Don't rely on execution)**
correctness	maintainability
reliability	flexibility & reusability
usability & performance	portability
completeness, consistency, usability	availability of correct & complete documents
robustness	structured & readability

Software Quality Attributes

Completeness refers to the availability of all features listed in the requirements, or in the user manual. An incomplete software is one that does not fully implement all features required.

Consistency refers to adherence to a common set of conventions and assumptions. For example, all buttons in the user interface might follow a common color coding convention. An example of inconsistency would be when a database application displays the date of birth of a person in the database in different formats ignoring user preference.

Difference of Term "Completeness" in Requirements and Testing Aspects?

Software Quality Attributes

Usability refers to the ease with which an application can be used. This is an area in itself and there exist techniques for usability testing. Psychology plays an important role in the design of techniques for usability testing.

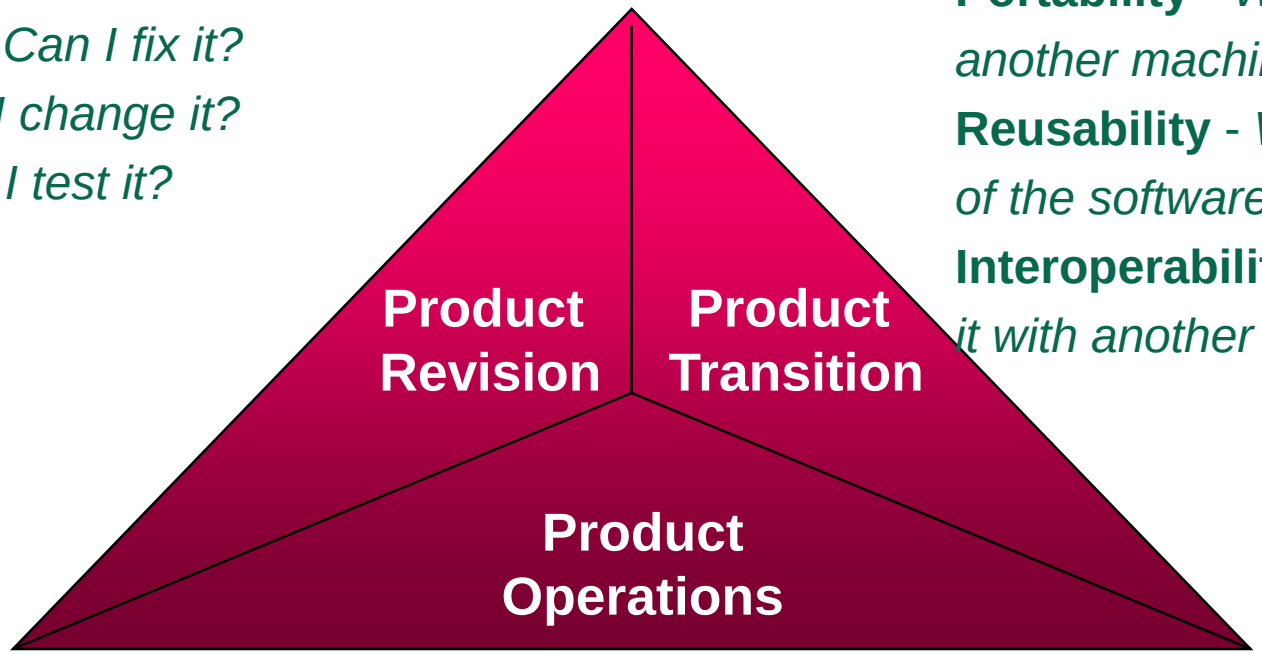
Performance refers to the time the application takes to perform a requested task. It is specified in terms such as ``This task must be performed at the rate of X units of activity in one second on a machine running at speed Y, having Z gigabytes of memory."

Non-functional requirement?

Software Quality Attributes

Maintainability - *Can I fix it?*
Flexibility - *Can I change it?*
Testability - *Can I test it?*

Portability - *Will I be able to use it on another machine?*
Reusability - *Will I be able to reuse some of the software?*
Interoperability - *Will I be able to interface it with another machine?*



-The **expected quality**

Features and characteristics

of a

software product

are commonly referred

to as **Quality Attributes**.

Correctness - *Does it do what I want?*
Reliability - *Does it do it accurately all the time?*
Efficiency - *Will it run on my machine as well as it can?*
Integrity - *Is it secure?*
Usability - *Can I easily use it?*

Quality People

- In large organizations or projects, how to ensure that the required level of product quality is achieved

Whose Responsibility?

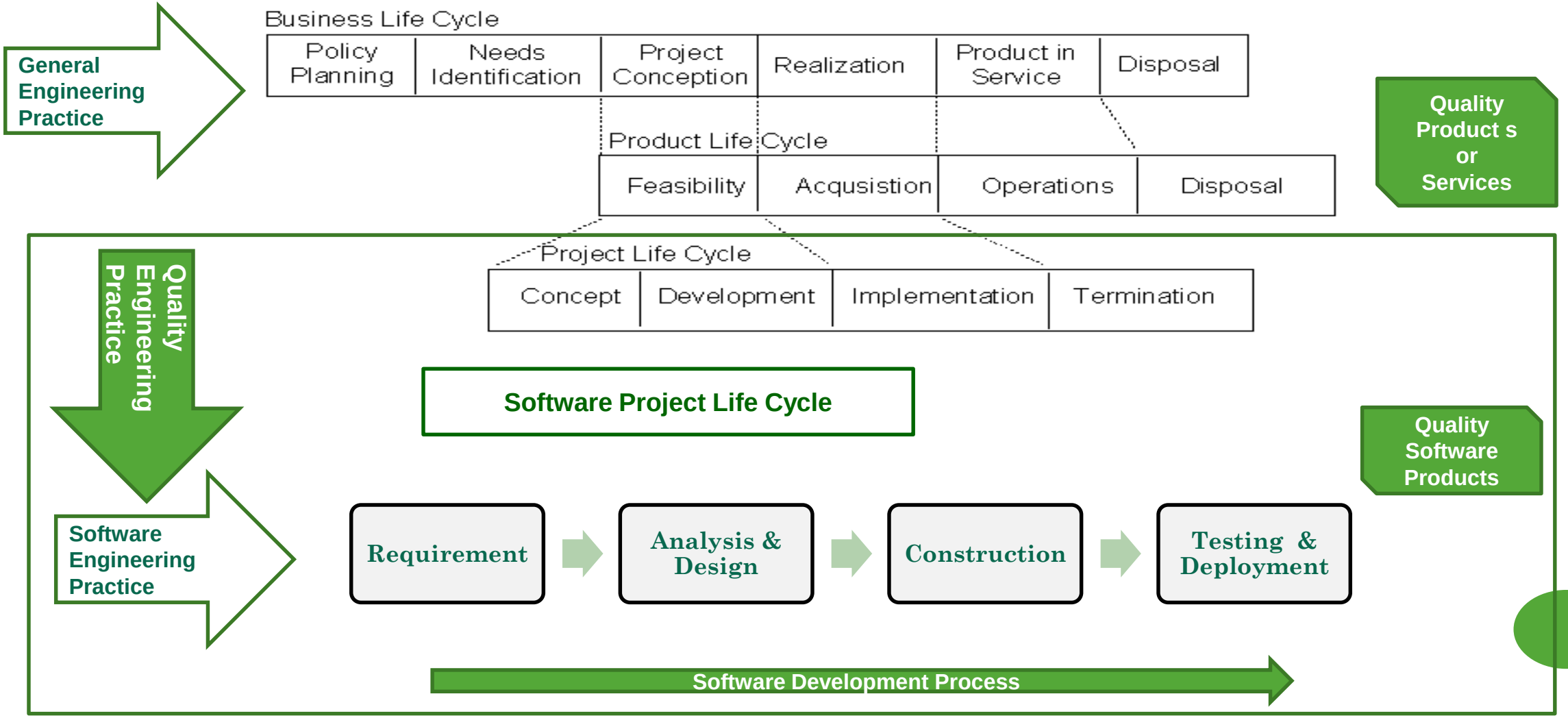
- the responsibility of the so-called quality managers, quality team / group, SQA, Tester etc.

- Quality group: Quality Coordinator director, manager, coordinator, members of different department i.e. HR, PM, Finance

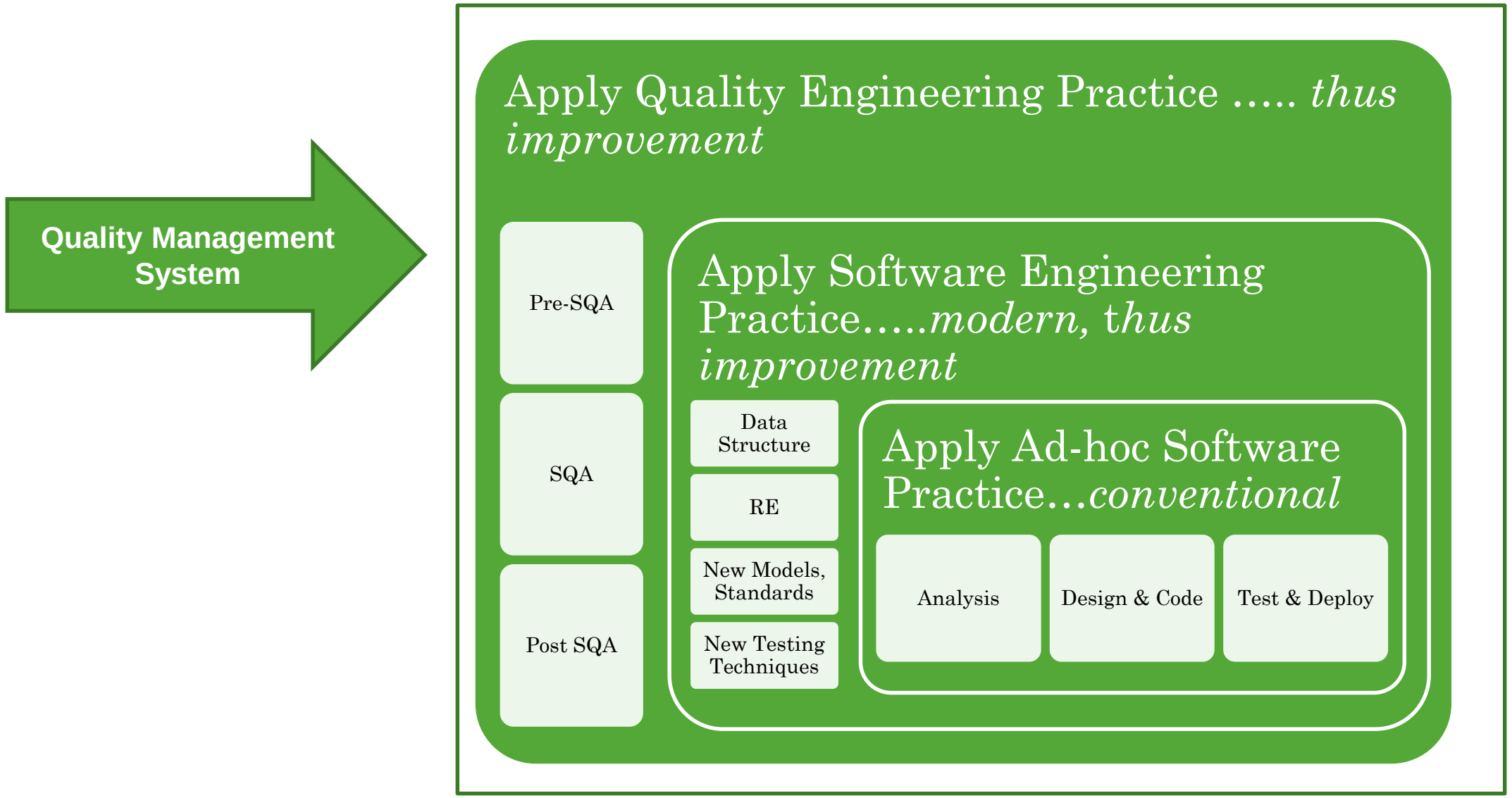
Terminologies w.r.t. Software Quality

- Software Quality Engineering SQE
- Software Quality Assurance SQA
- Software Quality Control SQC
- Software Testing (SQC by Only Testing)
- Total Quality Management TQM
- Quality Management System QMS
- Quality Improvement Process QIP
- How can you differentiate Software Testing with Software Quality*?*

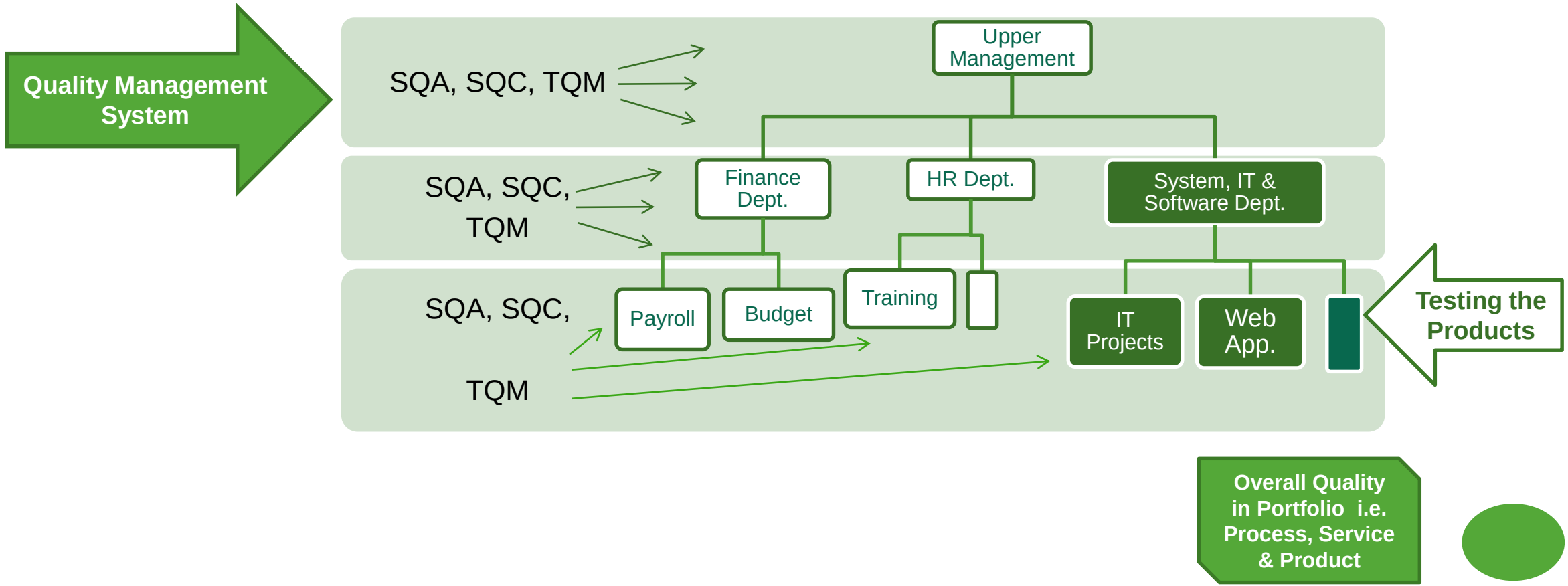
Apply Quality Engineering in Life Cycle?



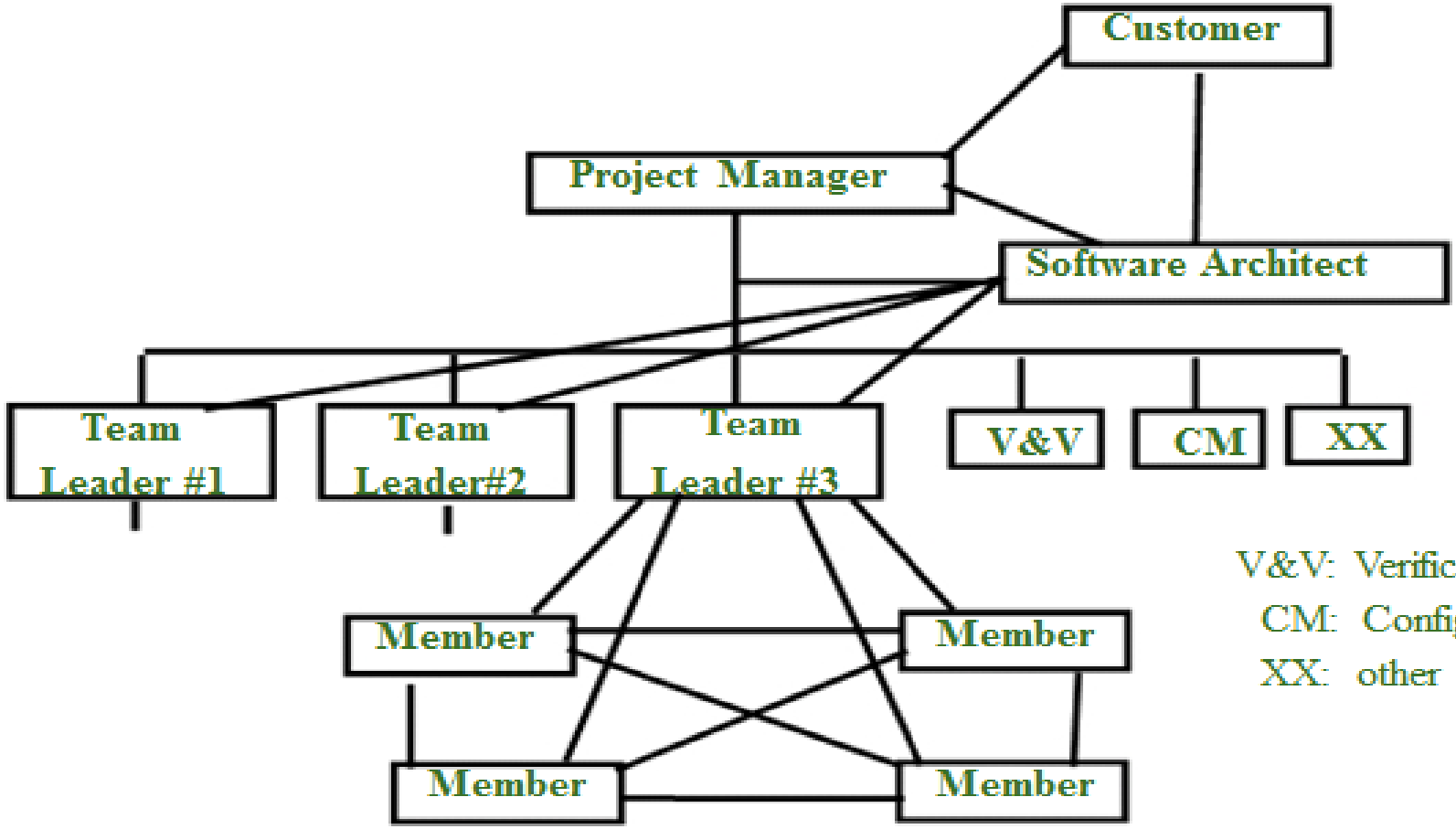
Overall Quality Approach



Apply Quality in Organization



An Team Structure for Software Developments

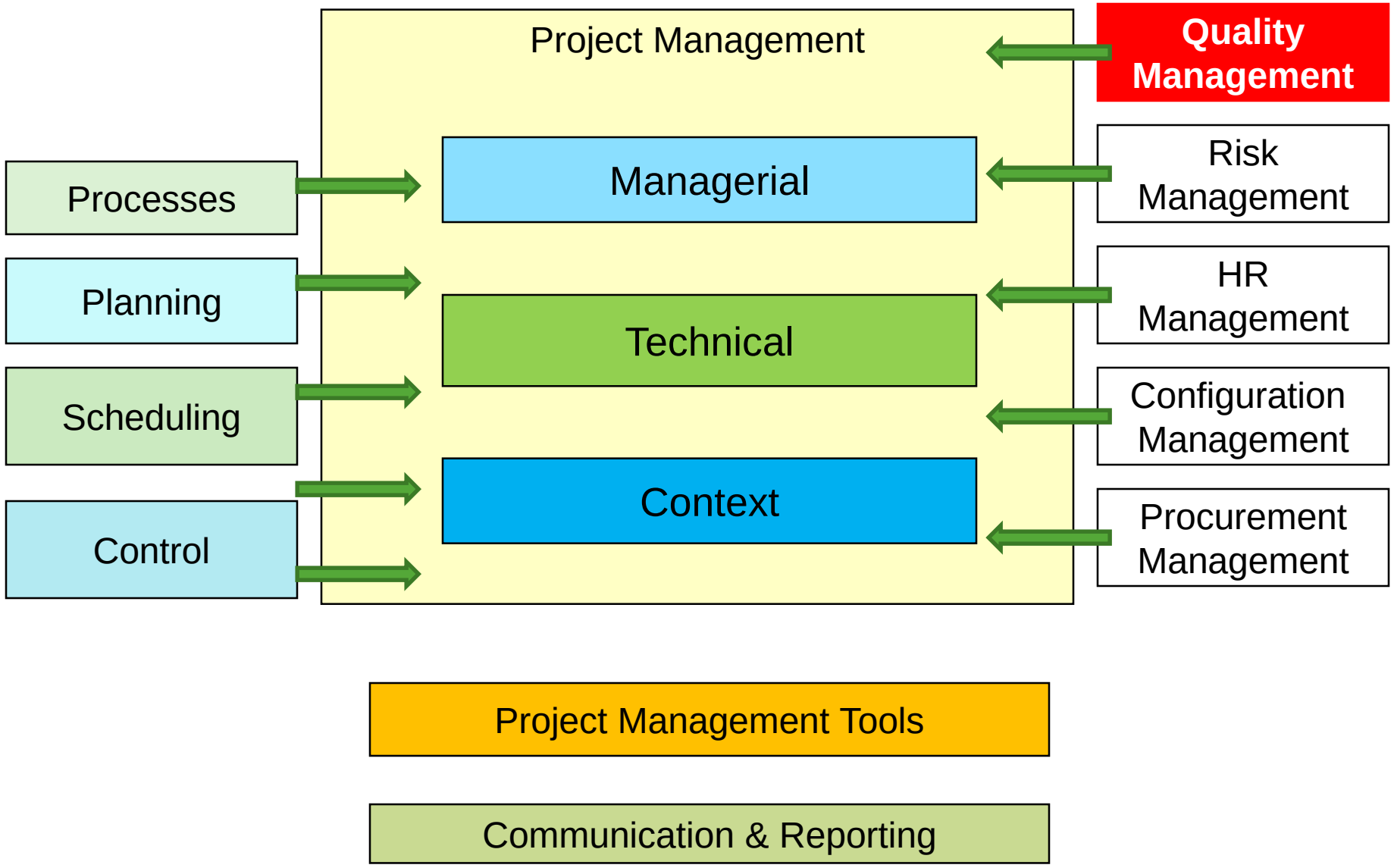


V&V: Verification and Validation
CM: Configuration Management
XX: other supporting processes

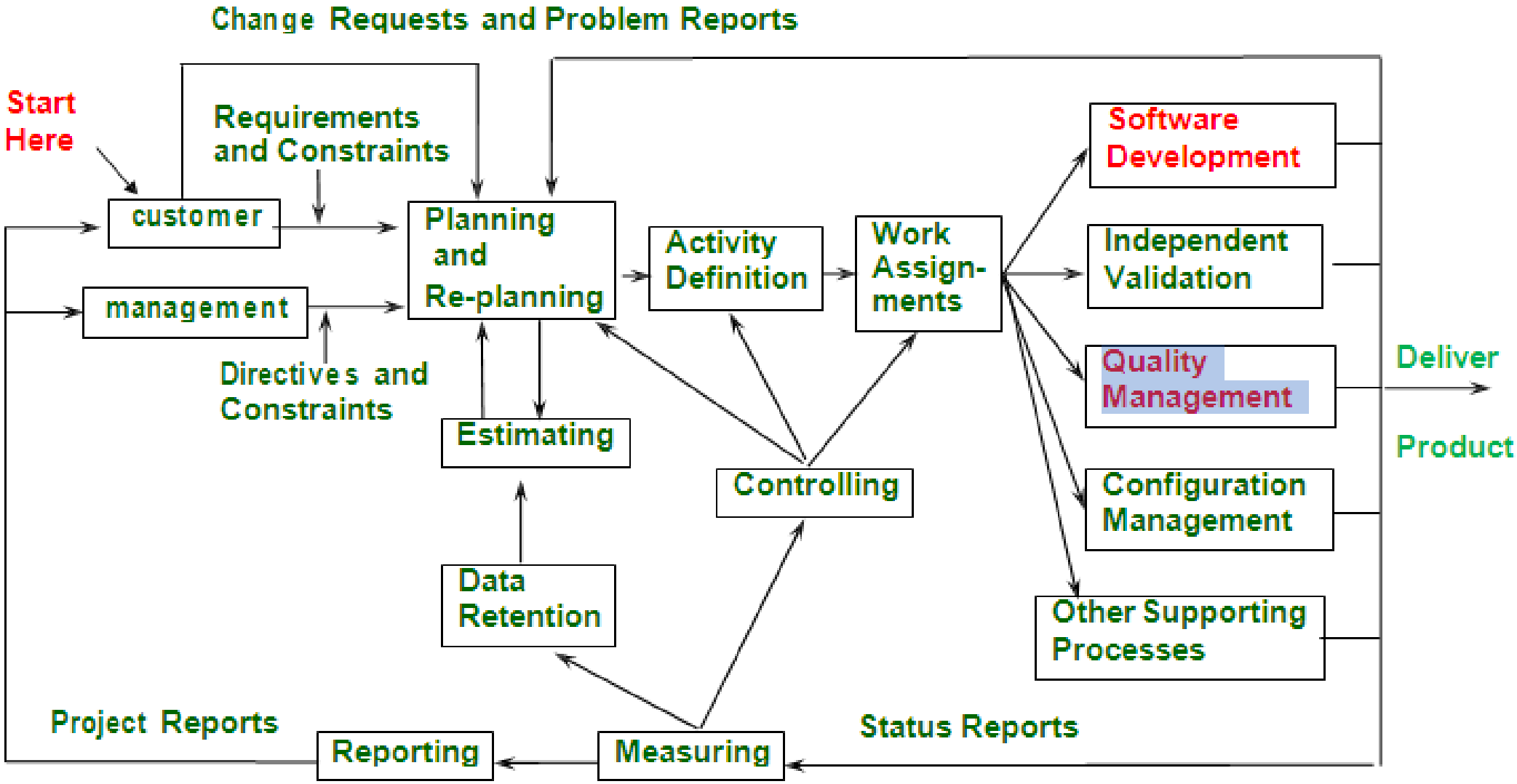
Each team has 2 to 5 members plus a team leader

Another View

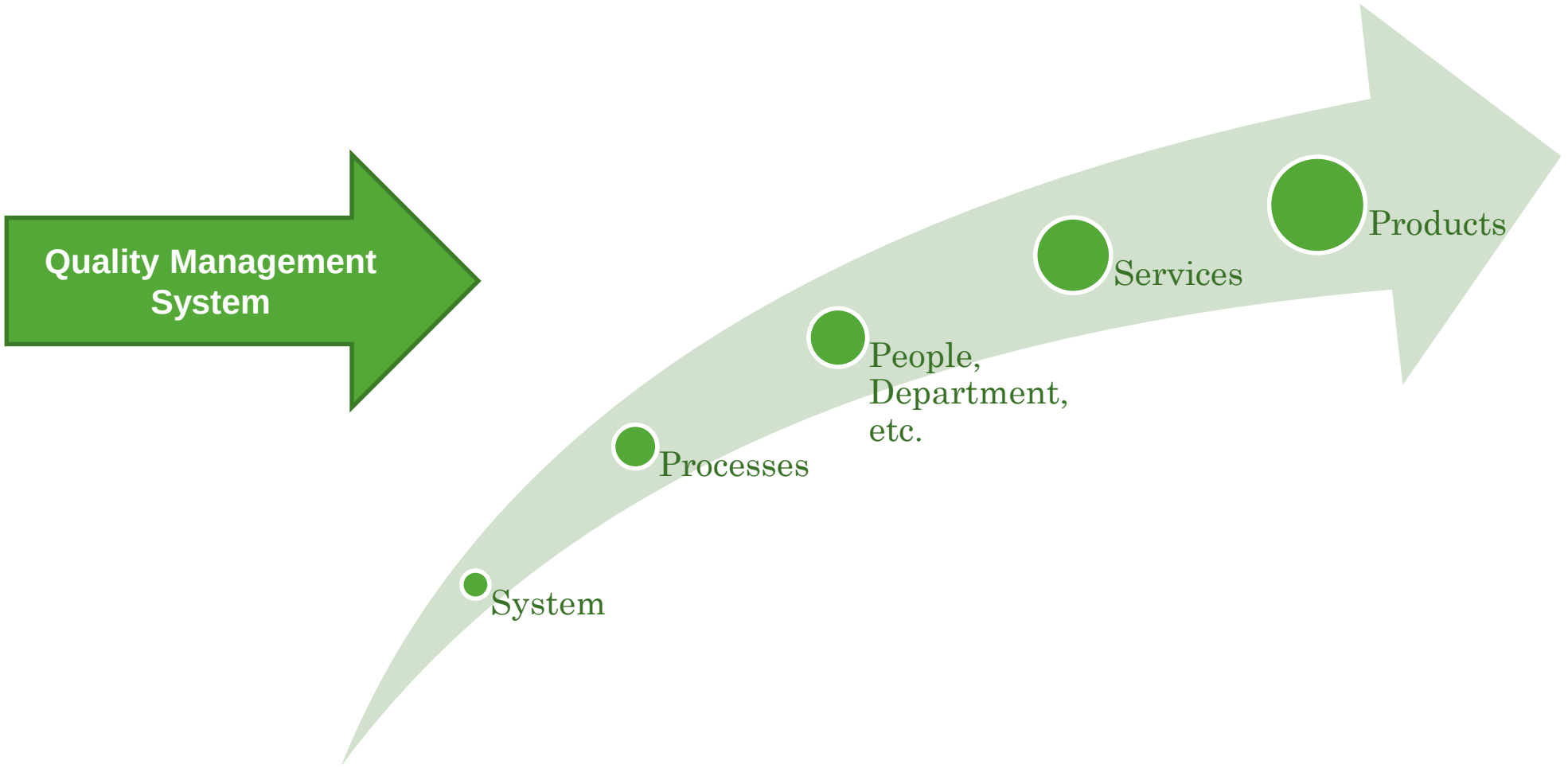
Quality Management – Supporting Processes



A Workflow Model for Software Projects



Overall Improvement

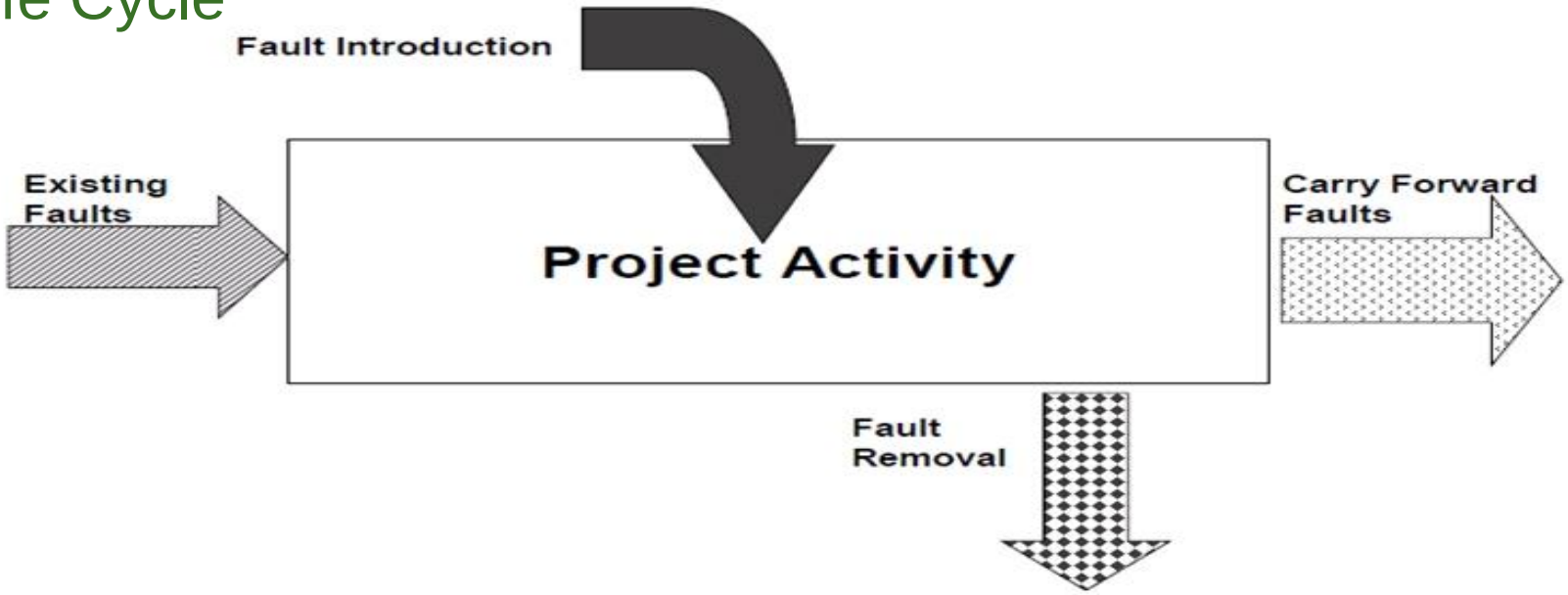


Software Quality

Software Quality Foundation

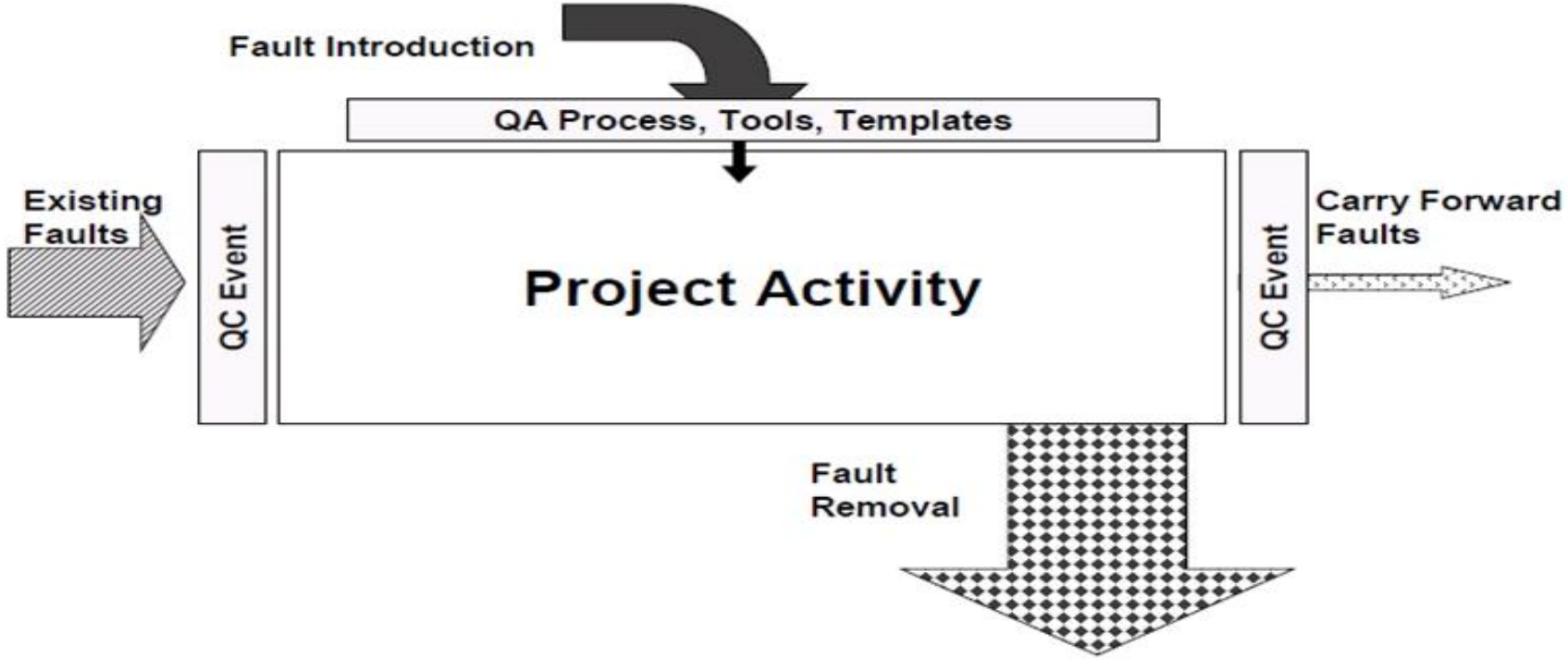
Software Lifecycle

- Defects are introduced in software by human activities through Software Life Cycle



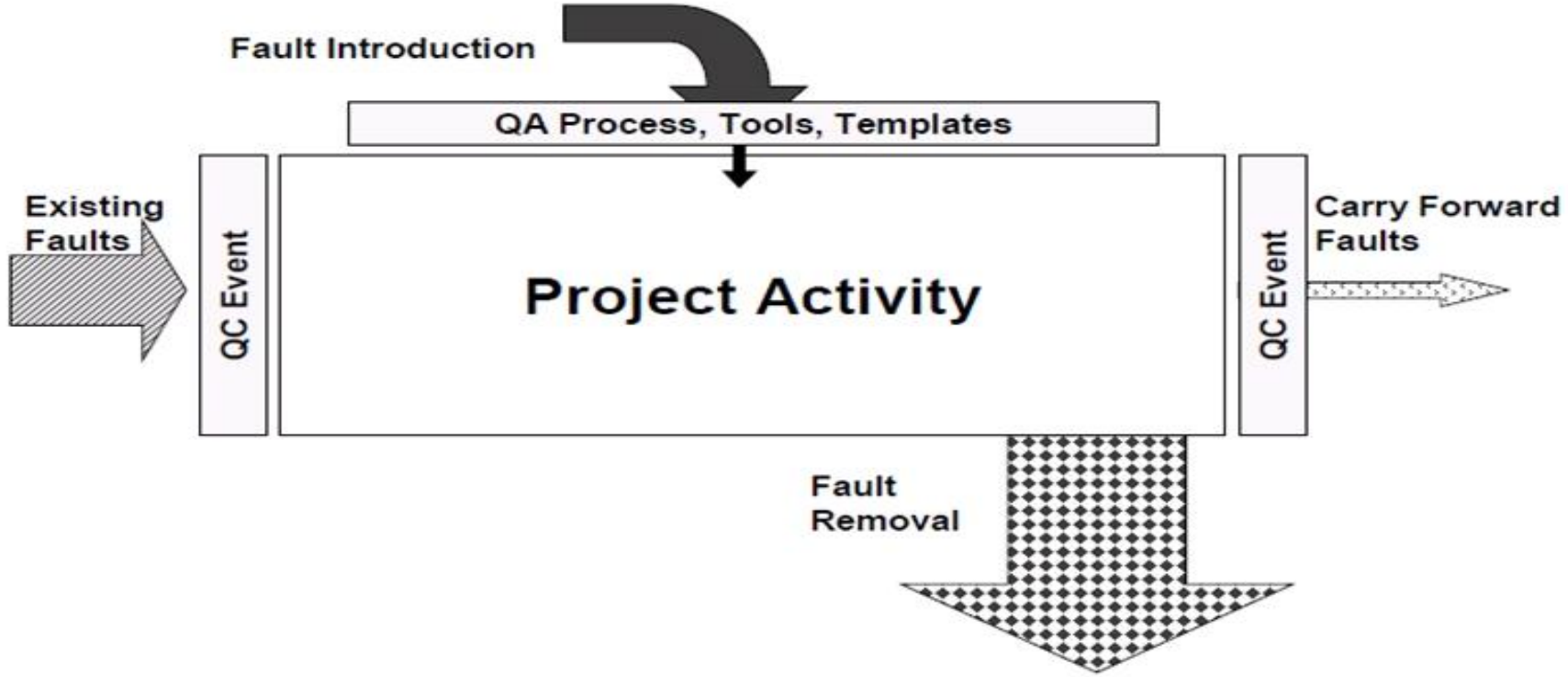
Software Quality

- **Quality Assurance (QA)** is fault **prevention** through process design and auditing
- Creating processes, procedures, tools, jigs, etc. to prevent faults from occurring
  - Examples: Templates, checklists, guides
  - **Main goal**: prevent as much as possible defect injection



Software Quality

- **Quality Control (QC)** is fault/failure **detection** through static and/or dynamic testing of artifacts
- Examining the artifact against pre-determined criteria to measure conformance
- **Main goal**: stop leakage of defects before deployment, find, fix, or containment



Difference in (QA) & (QC) ?

- Quality Assurance (QA)

- Things done before developing software & related produces

- Recruit Good People
- Develop Good Processes
- Provide Good Tools
- Train People to use Processes & Tools
- Provide Adequate Supervision
- Appreciate Good work
- Plan for QC

- Quality Control (QC)

- Things done after developing software & related products

- In spite our best efforts, some mistakes will occur
- Detect mistakes & ensures that quality standards / procedures are followed

Difference in (QA) , (QC) & (Testing) ?

- **Prevention process & detective process**
- **QA** –those activities which develop / modify processes to prevent introduction of defects
- **QC** –those activities which find & correct the flaws
- **QA Focus** ---- Process-oriented,

  Randomly Evaluate the product to confirm if process works,

  Ensures if process is defined & right, *preventing defect from occurring*

  e.g. Development of methodologies & standards: Review ***if requirement being defined at proper level of detail?***
- **QC Focus** ---- Products-oriented ,

  Continuous activity & observe if defective,

  Focus on finding, detecting & correcting the defects in specific deliverables

  e.g. Are defined requirements the right requirements?
- **Testing** ----- The process of executing a system with the intent of finding defects
- **Note:** Process of executing a system includes **test planning** prior to the execution of the test cases
- **Testing is one part of a QC activity, but there are others such as reviews, inspections etc..**