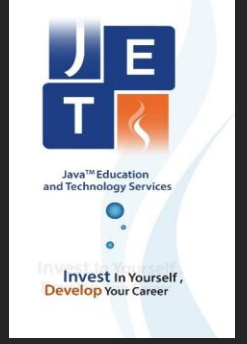


ITI - Intake 46

Introduction to Software Engineering

Beshoy adel

Evaluation Criteria

- Duration: 4 days (**2 lectures**).
- **30%** Attendance, Activities and Class Participation.
- **70%** Presentation&Project .

Topics Covered

- **Professional software development**

What is meant by software engineering and related topics.

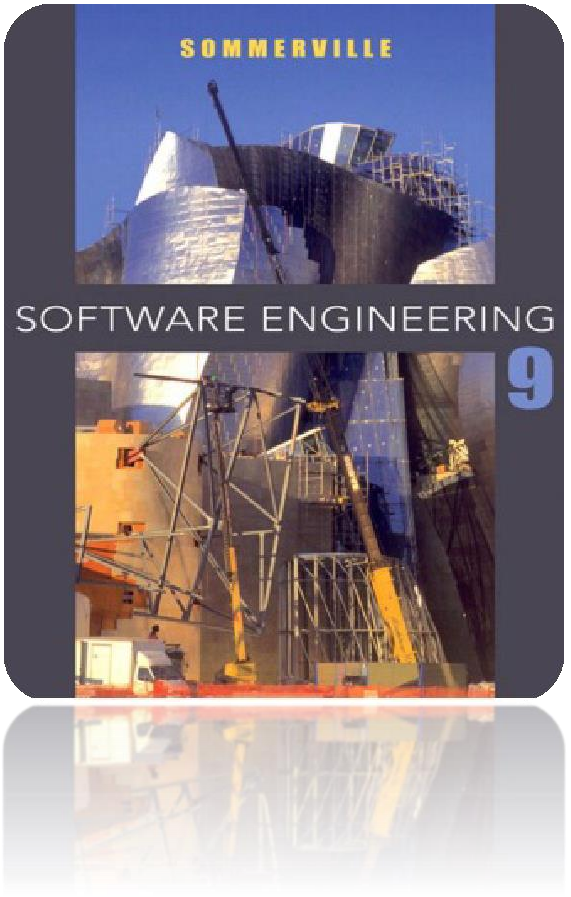
- **Software engineering ethics**

A brief introduction to ethical issues that affect software engineering.

- **Case studies**

An introduction to three examples that are used in later chapters in the book.

(SWE Ian Summerville 9 edition)



Agenda

DAY 1

- Program & Software
- Application types
- Good Software Attributes
- Software impact
- software Engineering
  - SWE Roles
  - SWE Skills
- SDLC

Agenda

DAY 2

- Software Testing
  - Error ,Bugs , Defect , Fault & failure
  - Testing stages and Types
  - Quality Assurance (QA)
  - Quality Control (QC)
- Software Models
  - Waterfall Model
  - Incremental Model
  - Spiral Model
  - Prototype
- Software engineering Ethics

“**Programming today is a race between software engineers striving to build bigger and better idiot-proof programs**, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning.”

—Rick Cook

Programming is NOT enough !

What is

software ??

Program

- **A program :** is a set of instructions executed by computer to perform a specific task.
- **Application:** A program or group of programs that is designed for the end user. Application software (an application) is a set of computer programs designed to allow users to perform a group of coordinated functions, tasks, or activities.

What is software?

Computer Programs

+

associated Documentation

Documentation Types

- User Documentation
- Technical Documentation
- Marketing Documentation

Software types

- Generic
- Customized

Application types

Activity Time

Application types

- ✓ Stand- Alone
- ✓ Interactive transaction-based
- ✓ Embedded Control Systems
- ✓ Entertainment Systems
- ✓ Systems for modeling and simulation
- ✓ Data Collection Systems
- ✓ System of systems

Application types

- **Stand-alone applications**
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- **Interactive transaction-based applications**
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- **Embedded control systems**
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

Application types

- **Entertainment systems**
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- **Systems for modeling and simulation**
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

Application types

- **Data collection systems**
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- **Systems of systems**
  - These are systems that are composed of a number of other software systems.

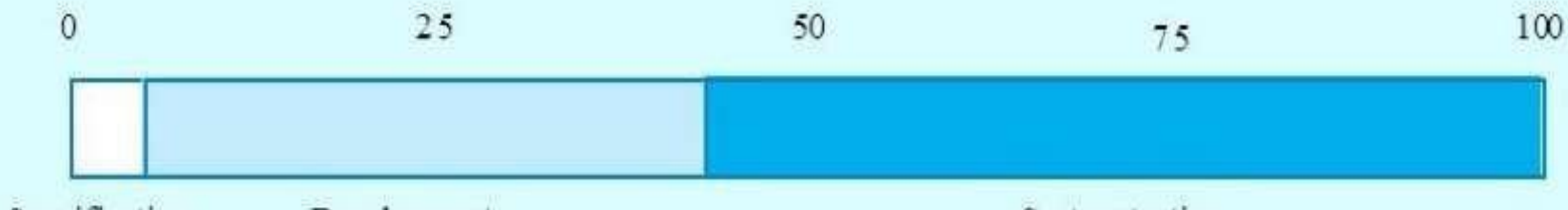
Attributes of Good Software

Attributes of Good Software

Software Cost

60% Development

40 % testing



Importante of software

Software can have a huge impact in any aspect of society.

Therac-25

```
PATIENT NAME: John
TREATMENT MODE: FIX          BEAM TYPE: E     ENERGY (KeV):      10

                             ACTUAL           PRESCRIBED
        UNIT RATE/MINUTE       0.000000         0.000000
        MONITOR UNITS        200.000000       200.000000
        TIME(MIN)              0.270000         0.270000


GANTRY ROTATION (DEG)          0.000000         0.000000        VERIFIED
COLLIMATOR ROTATION (DEG)    359.200000       359.200000        VERIFIED
COLLIMATOR X (CM)             14.200000        14.200000        VERIFIED
COLLIMATOR Y (CM)             27.200000        27.200000        VERIFIED
WEDGE NUMBER                   1.000000         1.000000        VERIFIED
ACCESSORY NUMBER               0.000000         0.000000        VERIFIED



DATE: 2012-04-16      SYSTEM: BEAM READY      OP.MODE: TREAT       AUTO
TIME: 11:48:58        TREAT: TREAT PAUSE              X-RAY       173777
OPR ID: 033-tfs3p     REASON: OPERATOR        COMMAND:
```

- Between 1985 and 1987, at least six patients died as a consequence of severe overdoses of radiation delivered.
- The cause was a fault in the control software.

Ariane 5 Flight 501 "1996"

Cause: design errors in the software

Video

Chaos Report 'Standish Group'

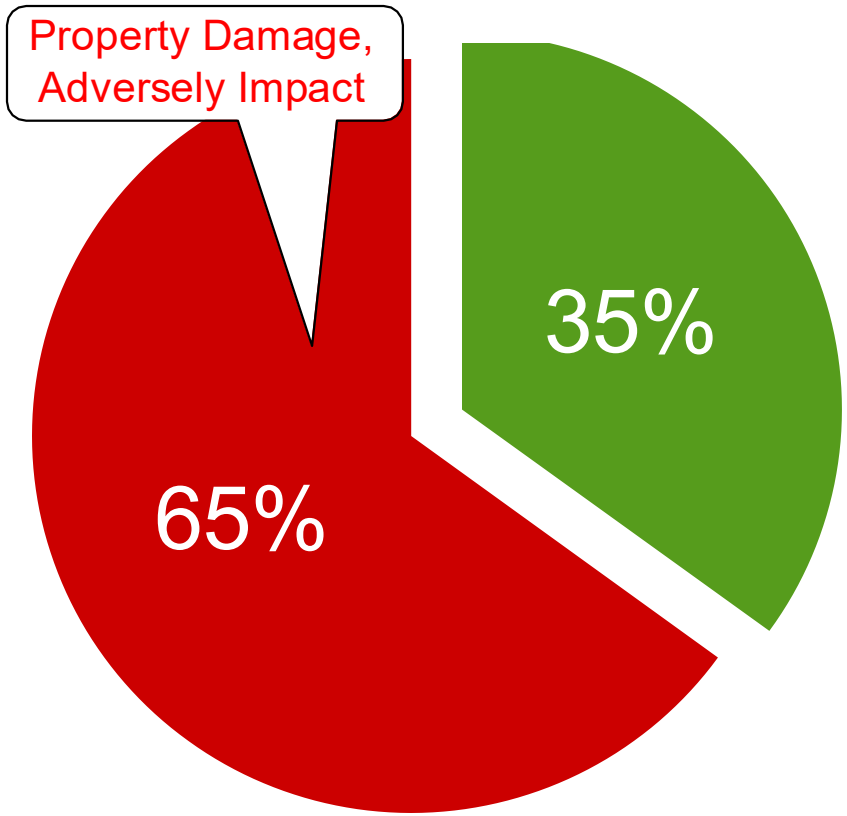
Chaos Report

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011-2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

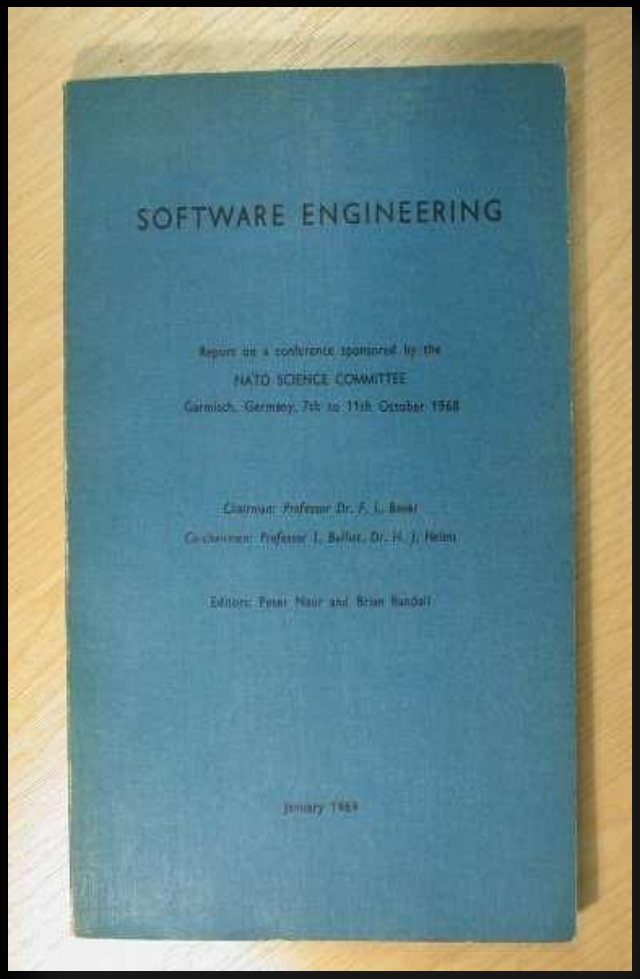
Most projects fail than succeed



- 65% Completed Late, Over Budget, Does not meet the re- quirements, Feature Missing, Not Easy to Use, Canceled
- 35% Successful

Projects fail !! WHY?

The origins of software engineering: NATO conference in 1968



the origin of software engineering and other fields

Θαλής ο Μιλήσιος
Thales of Miletus

Electrical Engineering

600 BC

3rd Century BC

Civil Engineering

Ἀρχιμήδης
Archimedes of Syracuse

George E. Davis

Chemical Engineering

1887

1752

Modern Electrical Engineering

Benjamin Franklin

NATO Conference

Software Engineering

1968

1954

Biological Engineering

Professor Heinz Wolff

Time

Now

The role of software engineering is to capture the customer's business needs and specify the "blueprints" for the system so that programmers can implement it

What is software engineering?

Software engineering:

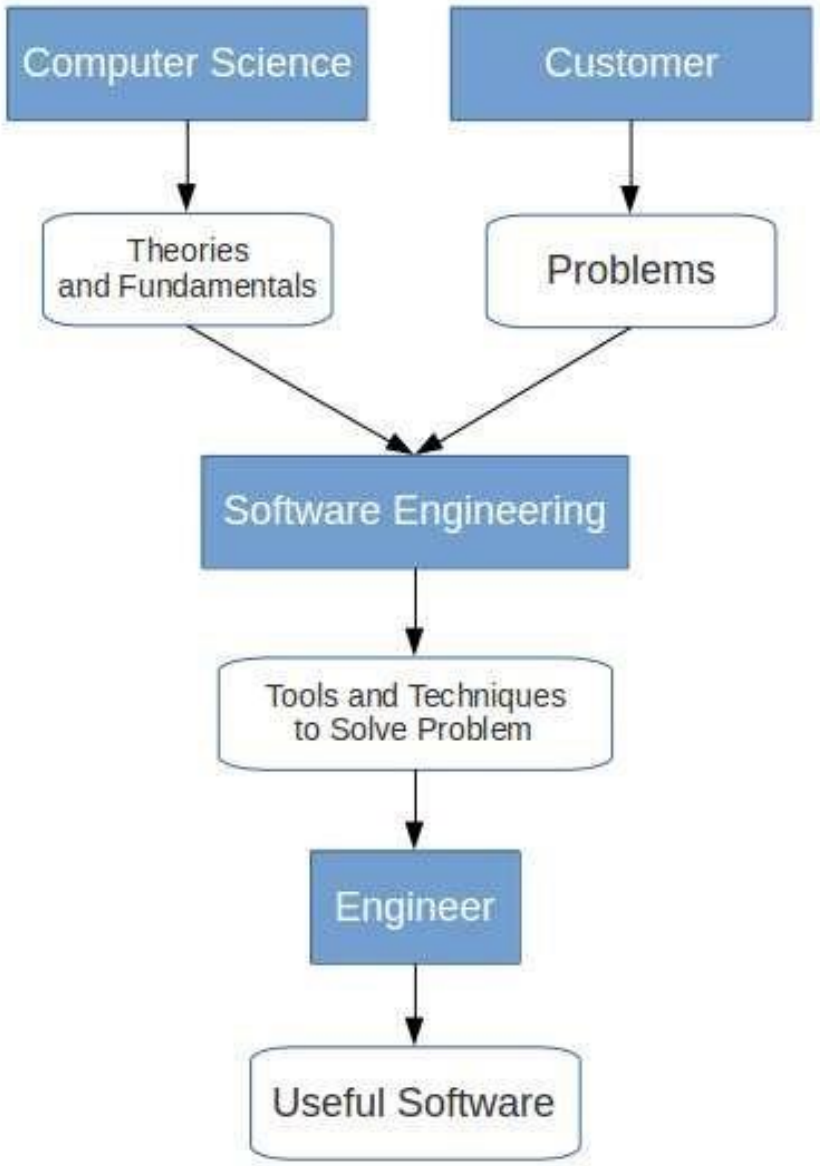
is an engineering discipline that is concerned with all aspect of software production.

Most effictive way
to produce hight-quality
software

Software engineering vs computer science

- Computer science focuses on theory and fundamentals.

- Software engineering is concerned with the practicalities of developing and delivering useful software.



The Team


Onsite PM
TEST Team
T Lead
P Manager
SOLID
BORN BRAVE
Developer
BUG

Daily Software Engineering Jobs/ Roles

1. Requirement Engineer
2. Systems Analyst
3. Designer
4. UI/UX developer
5. Software Programmer
6. Software Systems Administrator

7 Software Database Administrator

8 Software Security Engineer

9 Software Tester

10 Software Project Manager

11 Software Configuration Engineer

12 IT Help Desk

Position Levels

- Trainee / fresh
- Junior
- Mid level
- Senior
- Team Leader
- Project Manager

Software Engineering

Skills

Communication

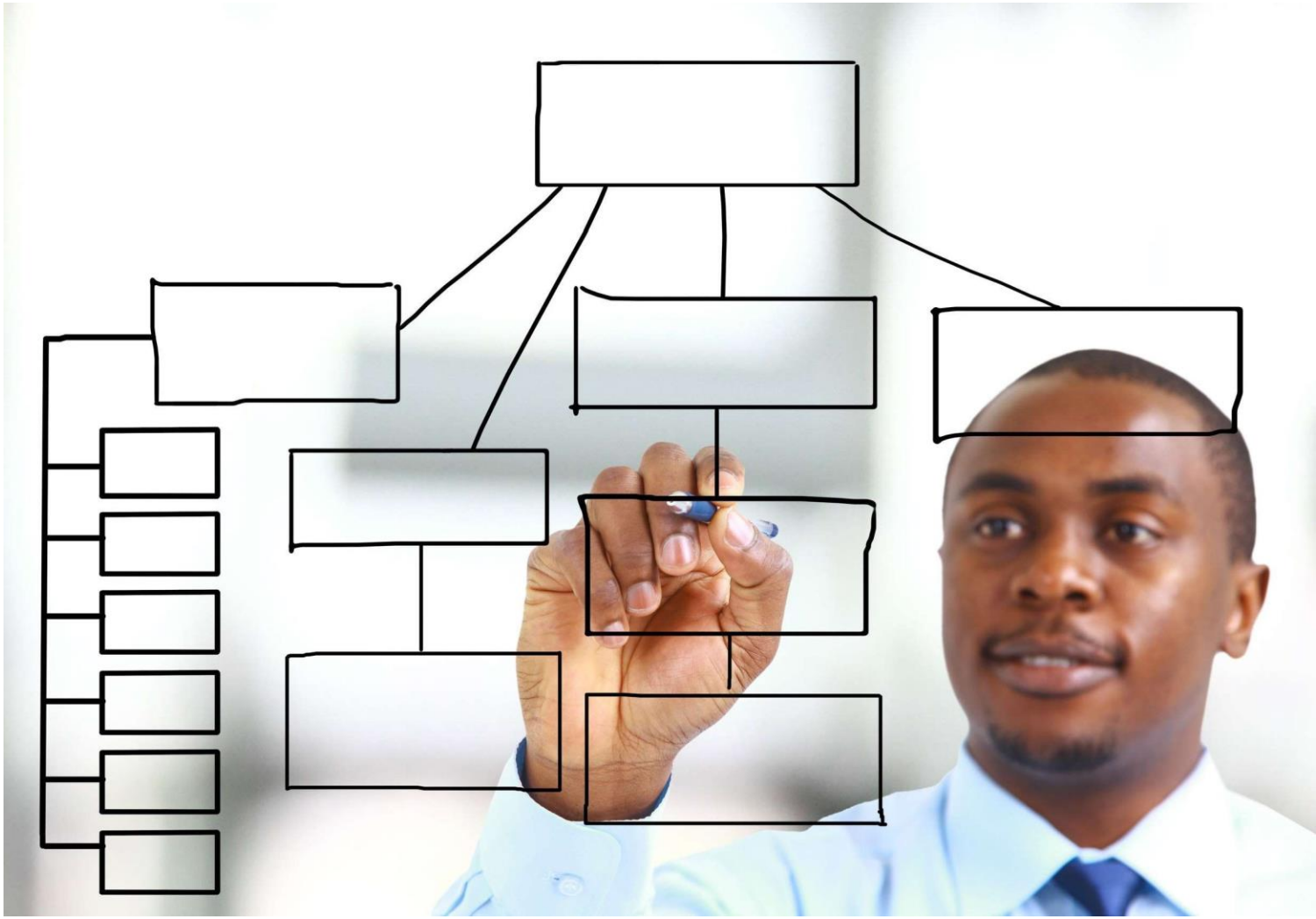
“There are only 10 types of people in this world.

Those who know binary, and those who don’t .”

Feedback

Motivation

Analytical



Project Management Skills

Planning

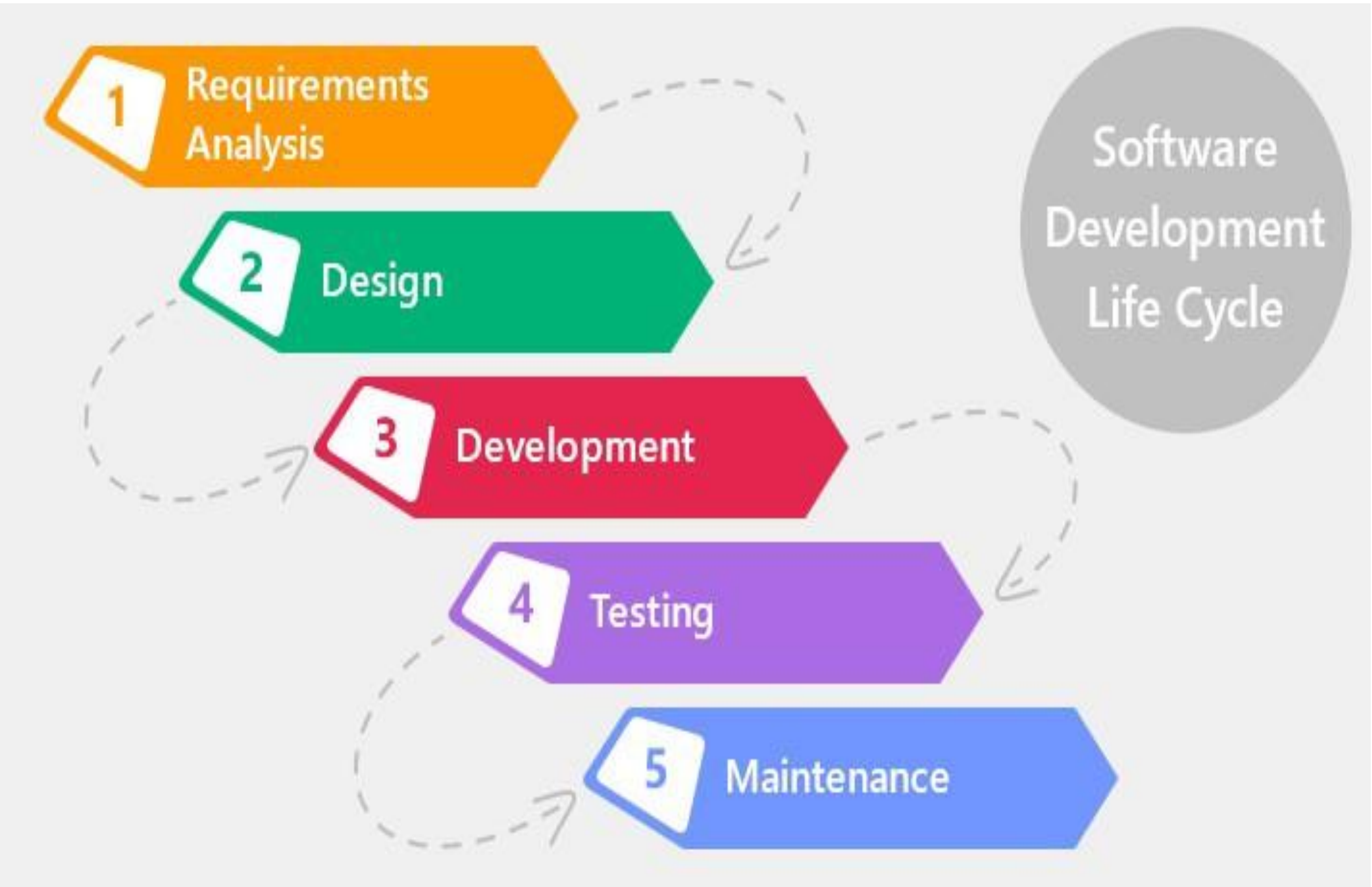
Risk Assessment

Estimation

Software Process

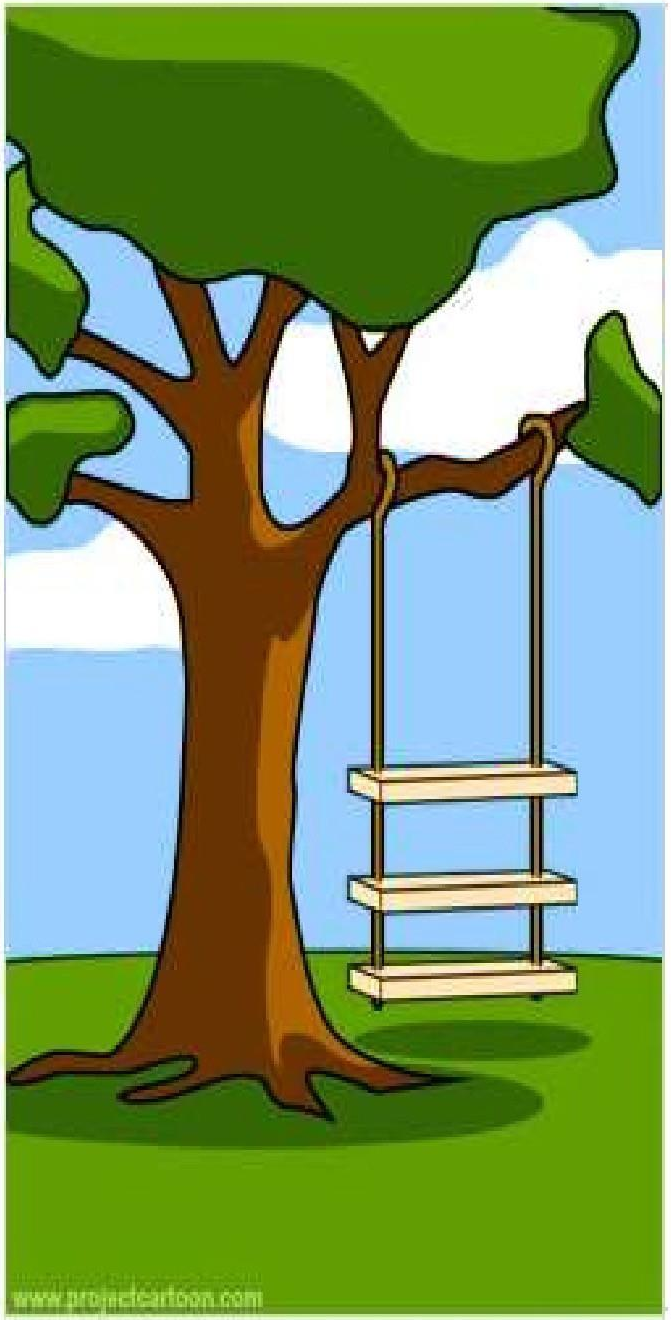
- A software process (also knows as software methodology) is a set of related **activities** that leads to the **production** of the software. These activities may involve the development of the software from the scratch or modifying an existing system.

Stages for software development



Requirement

- descriptions of what the system should do.
- These requirements reflect the needs of customers for a system that serves a certain purpose , it provides the constraints on its operation.



How the customer explained it

How the team designed it

What the customer really needed

Problems of requirements elicitation

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.

Requirements Engineering

- The process of finding out, analyzing, documenting and checking these services and constraints

Interviewing

interviews with stakeholders are part of most Reprocesses.

- Types of interview
  - ✓ Closed interviews based on pre-determined list of questions
  - ✓ Open interviews where various issues are explored with stakeholders.

- **Effective interviewing**
  - ✓ Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - ✓ Prompt the interviewee to get discussions going using a springboard question, a requirements proposal or by working together on a prototype system.

Interviews in practice

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviews are not good for understanding domain requirements

Scenarios

- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Ethnography

- A social scientist spends a considerable time observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Levels of description Requirements

- **User requirements** : are statements, in a natural language, of what services the system is expected to provide to system users.

Ex: “The System shall generate monthly management reports showing the Profit for each product prescribed by each Branch during that month.”

Levels of description Requirements

- **System requirements (SRS) :** are more detailed descriptions of the software system's functions, services, and operational constraints.
- **should define exactly what is to be implemented.**
- It may be part of the contract between the system buyer and the software developers.

System requirements

“1.1 On the last working day of each month, a summary of the products , their profit , and the branch shall be generated.

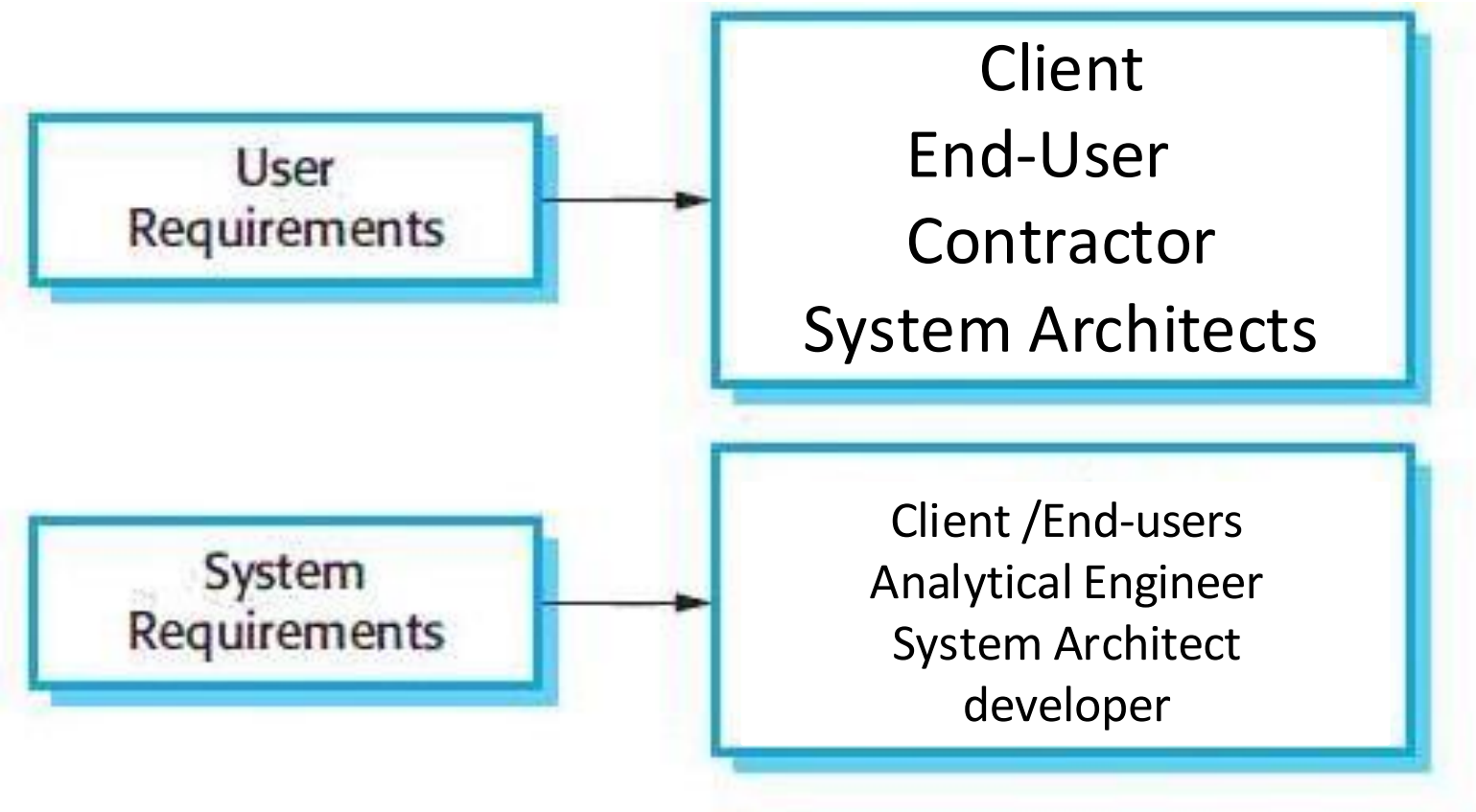
1.2 The system shall automatically generate the report for printing after 7.30 on the last working day of the month.

3. A report shall be created for each branch and shall list the individual Product names, the total number of this product , the number of sold, and the total Profit of this product .

4. If Product are available in Different sizes(e.g., 10 mg, 20 mg) separate reports shall be created for each Size unit.

5. Access to all profit reports shall be restricted to authorized users listed on a management access control list.”

Who Read it !!?


User Requirements
Client
End-User
Contractor
System Architects
System Requirements
Client /End-users
Analytical Engineer
System Architect
developer

System requirements classification

- ***Functional requirements*** *:* These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.
- ***Non-functional requirements*** *:* These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process

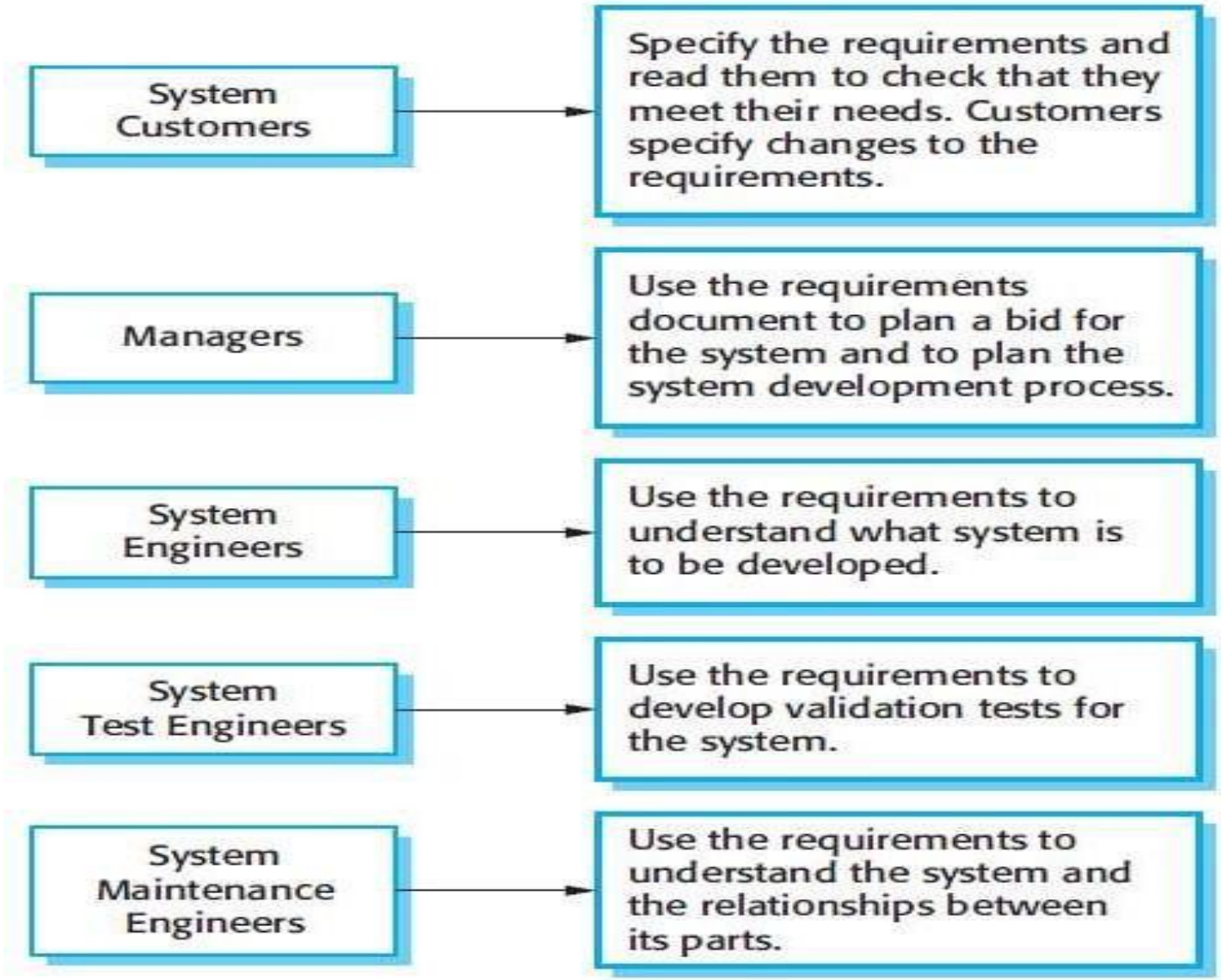
Types of non-functional requirement

- Access
- Usability
- Cost
- Capacity
- Documentation
- recovery
- Efficiency
- Backup
- licensing
- Performance
- Platform compatibility
- Privacy
- Quality
- Response time
- Security
- Support
- Test
- Speed

software requirements document

- sometimes called the software requirements specification or SRS
- is an official statement of what the system developers should implement.
- It should include both the user requirements for a system and a detailed specification of the system requirements.

Users of a requirements document



The structure of a requirements document

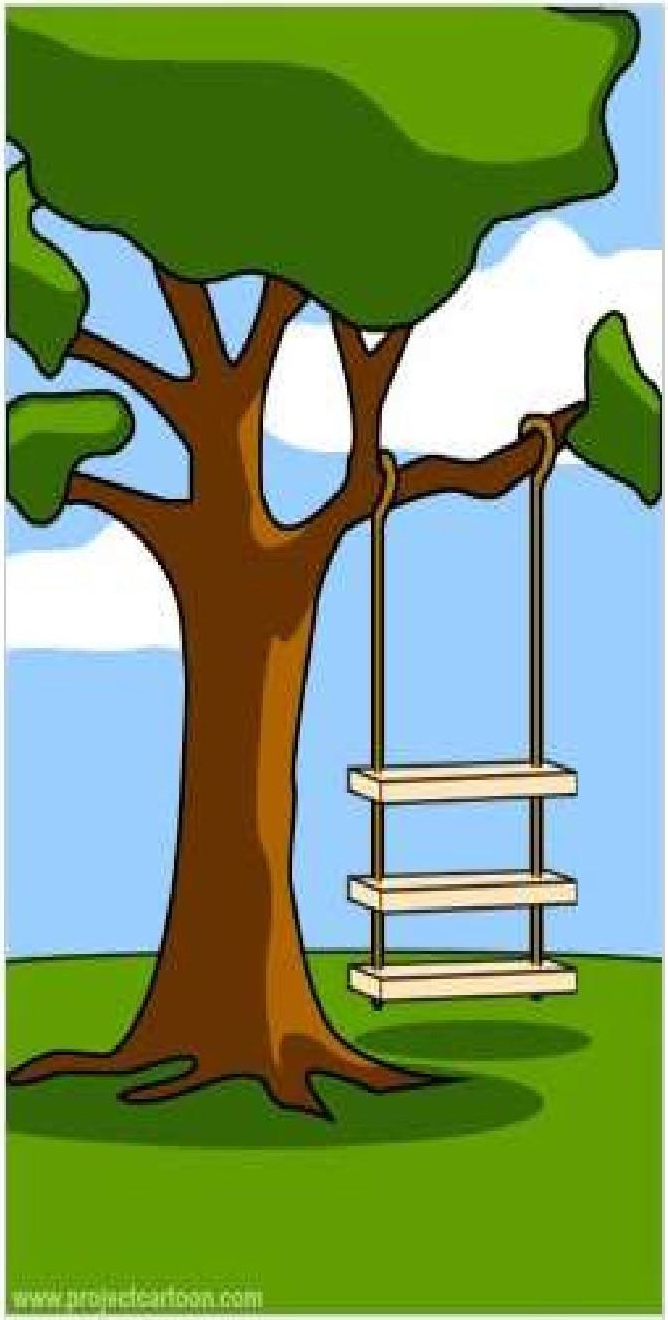
Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.



How the customer explained it

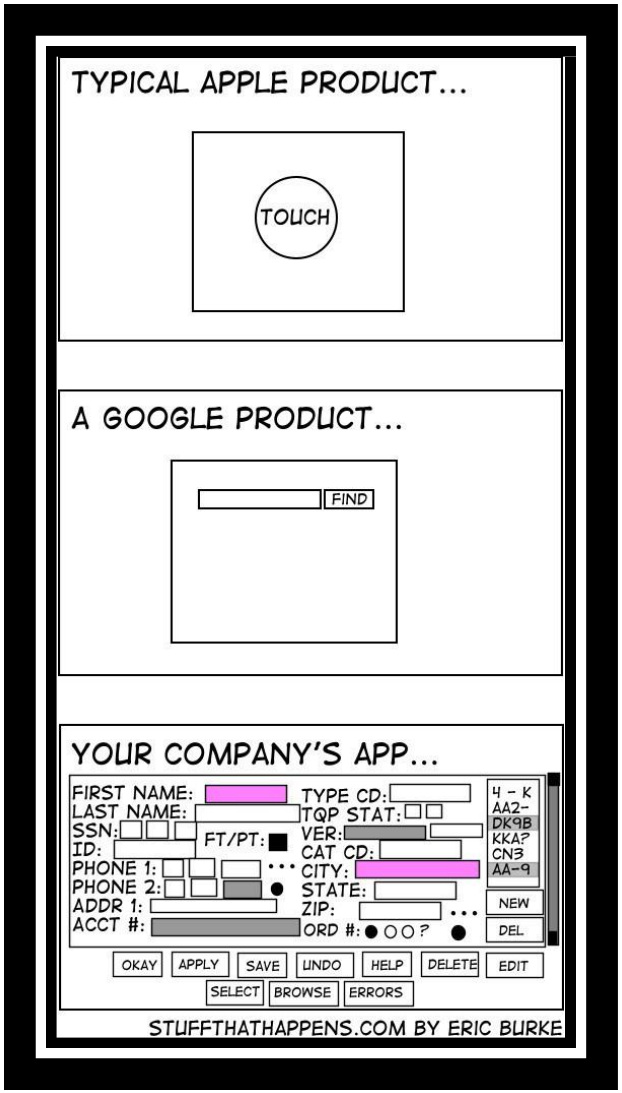
How the team designed it

What the customer really needed

Requirements checking

- Does the system provide the functions which best support the customer's needs?
- Are there any requirements conflicts?
- Are all functions required by the customer included?
- Can the requirements be implemented given available budget and technology?

2. Design



Planning the software solution

Software Design

Software design is an iterative process through which requirements are translated into a blueprint for constructing the software.

- Initially, the blueprint depicts a holistic view of software.
- During the design process the software specifications are transformed into design models .
- Models describe the details of the data structures, system architecture, interface, and components.

Modularity

- In this concept, software is divided into separately named and addressable components called modules
- Follows “divide and conquer” concept, a complex problem is broken down into several manageable pieces
- Easier to change
- Easier to build
- Easier to maintain

Interface Design

- UI is the user interface. This comprises everything a user can see and touch, such as menu options, buttons, text, layouts, navigation elements, sharing options, etc

- UX is why you made that change to affect how the user feels and behaves. The user experience is an umbrella term for the user's overall experience with the product: what they liked about it and how easily they accomplished their goals

Interface Design

UI

UX

Interface Design

UI

UX

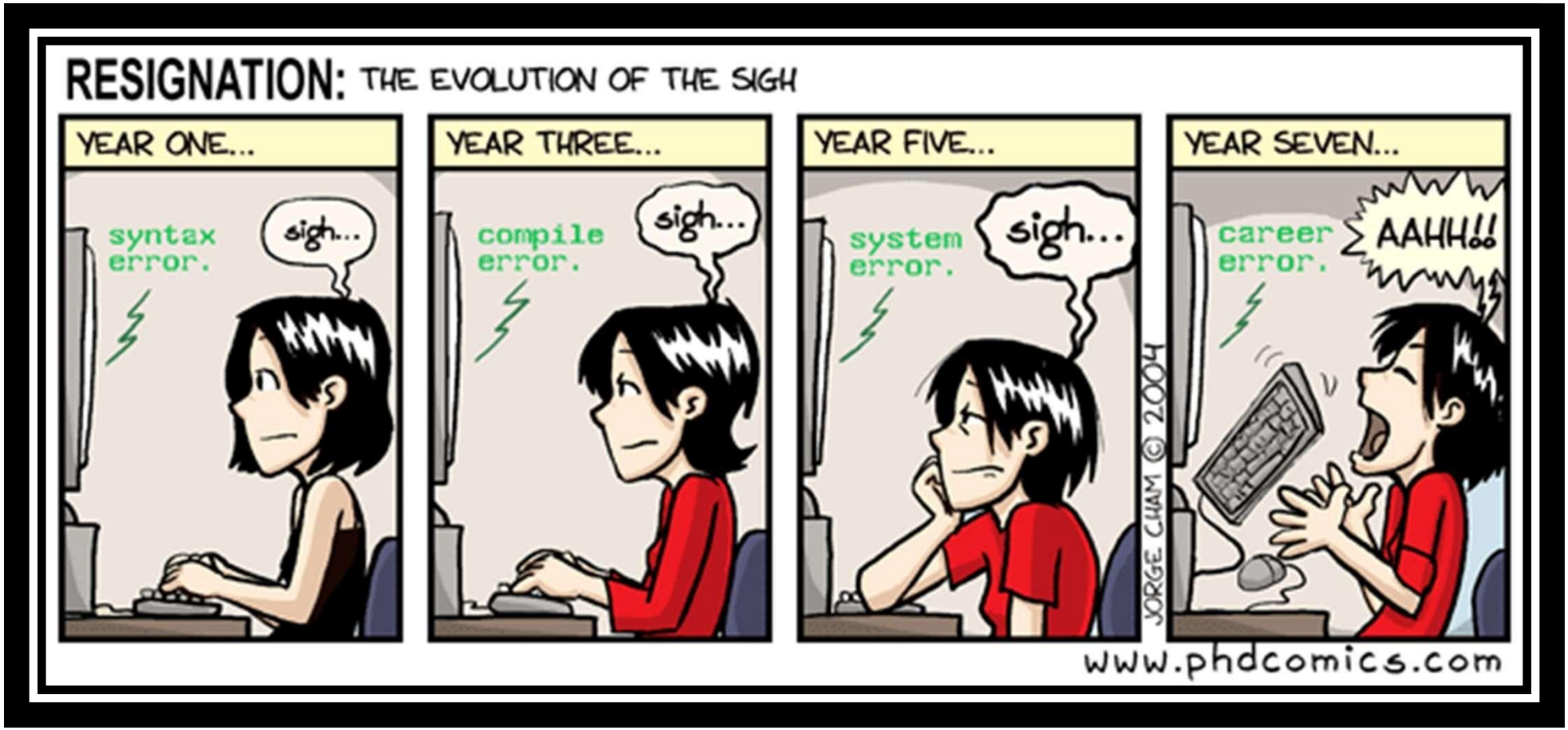
Open top, shake vigorously

Pray it doesn't get all over you

Open top

Squeeze out desired amount

3. Implementation



Code!!!

Refactoring

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure.

When software is refactored, the existing design is examined for

- redundancy
- unused design elements
- inefficient or unnecessary algorithms
- poorly constructed or inappropriate data structures

or any other design failure that can be corrected to yield a better design.

Code Review

- Code review is a software quality assurance process in which software's source code is analyzed manually by a team or by using an automated code review tool.
- The Goal is purely to improving code quality.
- Reviewing the codebase makes sure that every software or new feature developed within the company is of high quality.

4. Testing

Executing the application trying to find software bugs

Verification vs Validation

- Verification:

  "Are we building the product right".

The software should conform to its specification.

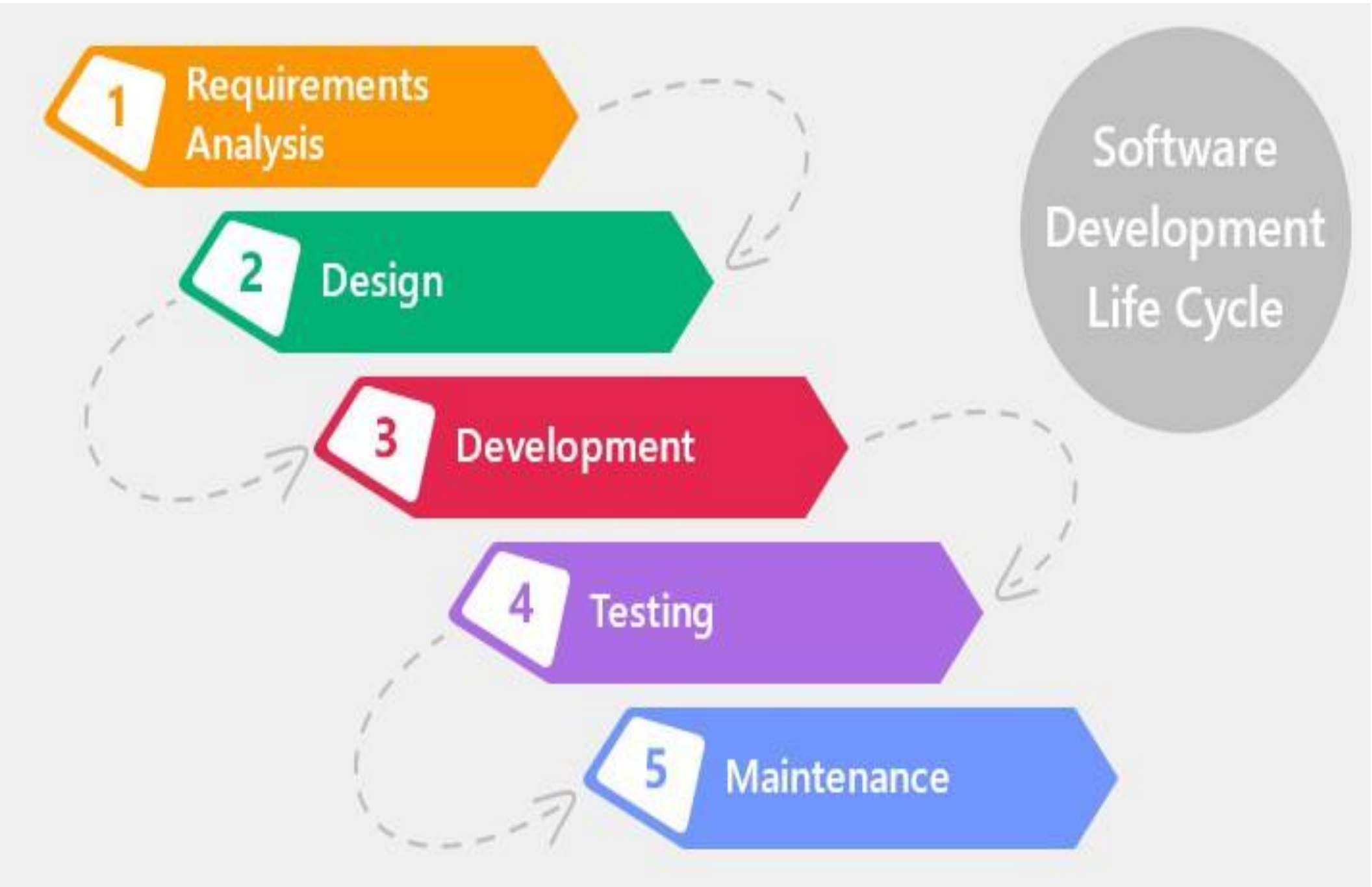
- Validation:

  "Are we building the right product".

The software should do what the user really requires.

TO BE
CONTINUED...

Stages for software development



Software Testing

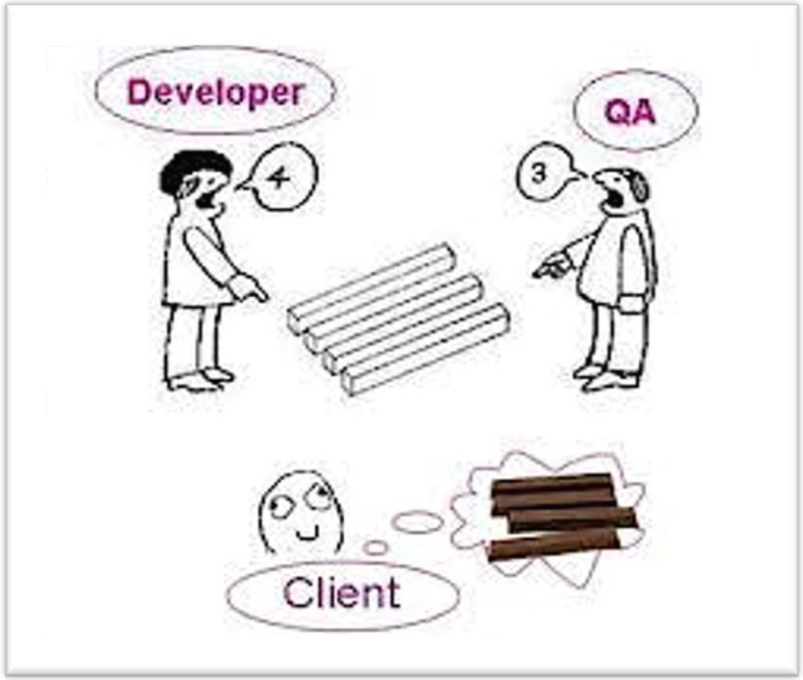
Software Testing

- 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.
- When you test software, you execute a program using artificial data.
- Testing is part of a more general verification and validation process, which also includes static validation techniques

Why Software Testing ?

Software Testing is important as it may cause mission failure, impact on operational performance and reliability if not done properly.

Effective software testing delivers quality software user's products satisfying requirements, needs and expectations.



Who Should Test?

- Developer
  - Understands the system
  - But, will test gently
  - And, is driven by deadlines
- Independent tester
  - Must learn system
  - But, will attempt to break it
  - And, is driven by "quality"

ERROR,
Bug,
Defect,
Fault & Failure

Bug, Fault & Failure

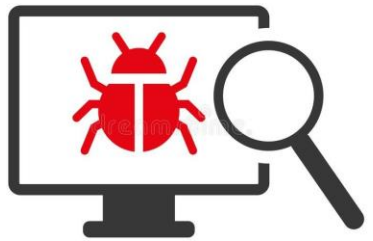
Error: An error is a human action that produces the incorrect result that results in a fault.

Bug: An Error found in the development environment before the product is shipped to the customer.

Defect : is the difference between expected and actual result in the context of testing.

Fault: A wrong or mistaken step, process or Data definition in a computed program which causes the program to perform in an unintended or unanticipated manner.

Failure: Deviation of the software from its expected result. It is an event.



Test Case

- A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.
- The process of developing test cases can also help find problems in the requirements or design of an application.

Stages of testing

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

Development testing

- Development testing includes all testing activities that are carried out by the team developing the system.
  - Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
  - Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.
  - System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

Release testing

- Release testing is the process of testing a particular release of a system that is intended for use **outside of** the development team.
- The primary goal of the release testing process is to convince the supplier of the system that it is good enough for use.
- Release testing is usually a black-box testing process where tests are only derived from the system specification.

Requirements based testing

- Requirements-based testing involves **examining** each **requirement** and developing a test or tests for it.
- In requirements-based testing, test cases are designed based on test objectives and test conditions that are derived from requirements.
- Examples of requirements-based tests are those that exercise specific functions or probe nonfunctional attributes, such as reliability or usability.

Scenario testing

- Scenario testing is an approach to release testing where you devise typical scenarios of use and use these to develop test cases for the system.
- A scenario is a story that describes one way in which the system might be used.
- Scenarios should be realistic and real system users should be able to relate to them.
- If you have used scenarios as part of the requirements engineering process , then you may be able to reuse these as testing scenarios.

Performance testing

- Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.
- Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable.
- Stress testing is a form of performance testing where the system is deliberately overloaded to test its failure behaviour.

Acceptance testing(User Testing)

A formal test conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.

It is the final test action before deploying the software. The goal of acceptance testing is to verify that the software is ready and can be used by the end user to perform the functions for which the software was built.

Alpha testing

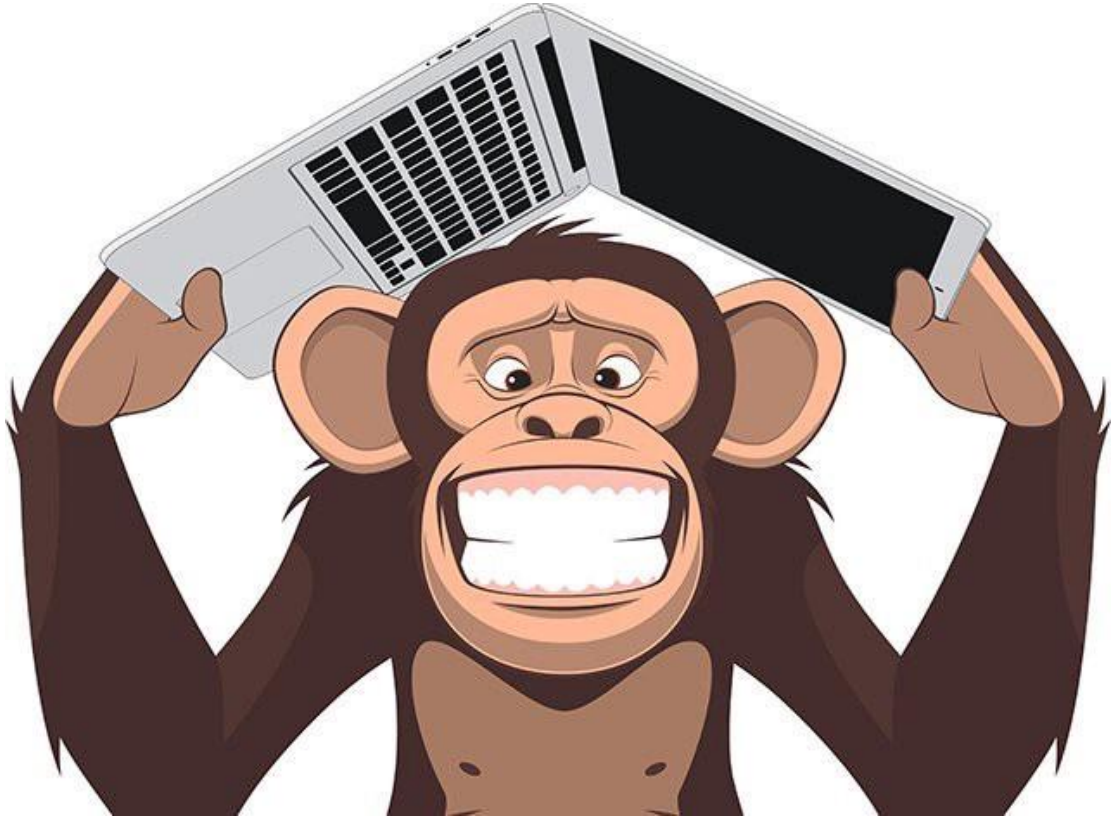
1. The application is tested by the users who doesn't know about the application.
2. Done at developer's site under controlled conditions
3. Under the supervision of the developers.

Beta testing

1. This Testing is done before the final release of the software to end-users.
2. Before the final release of the software is released to users for testing where there will be no controlled conditions and the user here is free enough to do what ever he wants to do on the system to find errors.

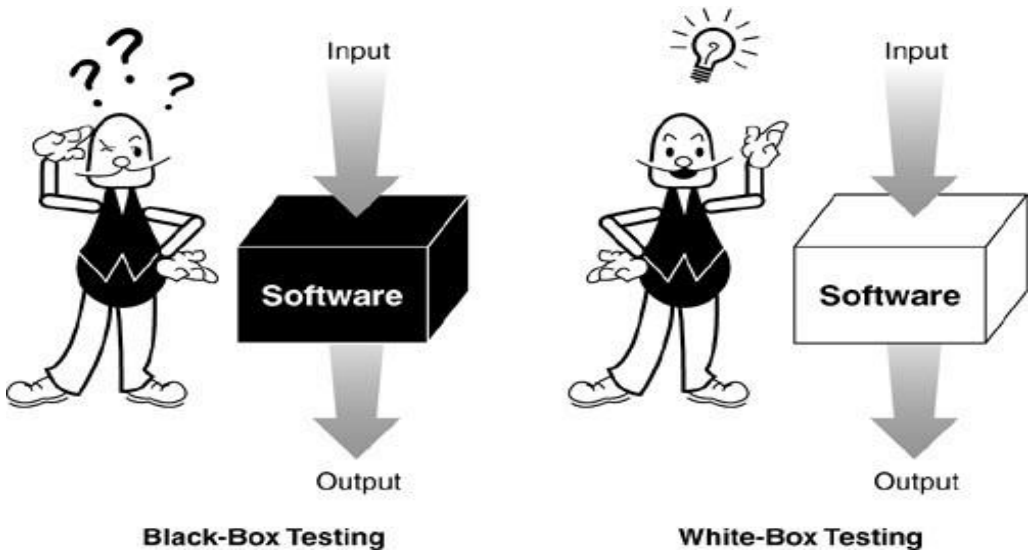
Monkey Testing

Testing the application randomly like hitting keys irregularly and try to breakdown the system there is no specific test cases and scenarios for monkey testing.



White Box vs Black Box

- White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
- Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester



Static Test vs Dynamic Test

- Under **Static Testing** code is not executed. Rather it manually checks the code, requirement documents, and design documents to find errors.

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

Bug Reporting Format

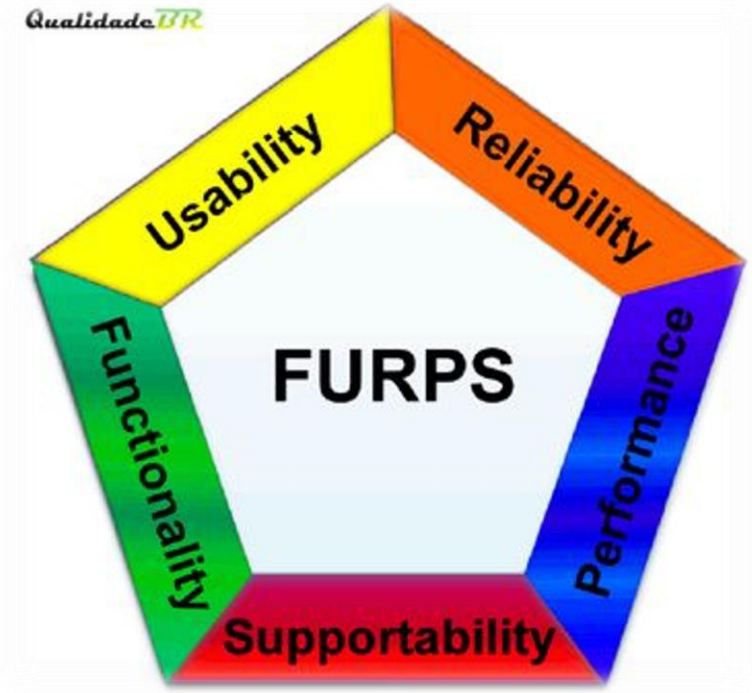
Bug Id	:	Sys_def_xyz_o1
Test case Id	:	Sys_xyz_o1
Bug Description	:	Specify the defect.
Bug Priority	:	Major or Minor
Identified by	:	ABC
Module	:	Database Operations
Project	:	XYZ Corporation
Dated	:	22/9/03

What is Quality ?

Quality is defined as meeting the customer's requirements and according to the standards

The best measure of Quality is given by FURPS

- **Functionality**
- **Usability**
- **Reliability**
- **Performance**
- **Supportability**



Quality Assurance

QA is a set of activities for ensuring quality in the processes by which products are developed.

it aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.

The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.

Quality Control

QC is a set of activities for ensuring quality in products. The activities focus on identifying defects in the actual products produced.

it aims to identify (and correct) defects in the finished product. Quality control, therefore, is a reactive process.

The goal of QC is to identify defects after a product is developed and before it's released.

QA vs. QC

- Quality Assurance makes sure that we are doing the right things, the right Way.
- QA focuses on building in quality and hence preventing defects.
- QA deals with process.
- QA is for entire life cycle.
- QA is managerial process.

- Quality Control makes sure the results of what we've done are what we expected .
- QC focuses on testing for quality and hence detecting defects.
- QC deals with product.
- QC is for testing part in SDLC.
- QC is corrective process.

5. Maintenance

Any activity oriented to change an existing software product.

Software processes

Software process is complex, it relies on making decisions. There's no ideal process and most organizations have developed their own software process.

Software Process Models

A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.

Plan-driven and agile processes

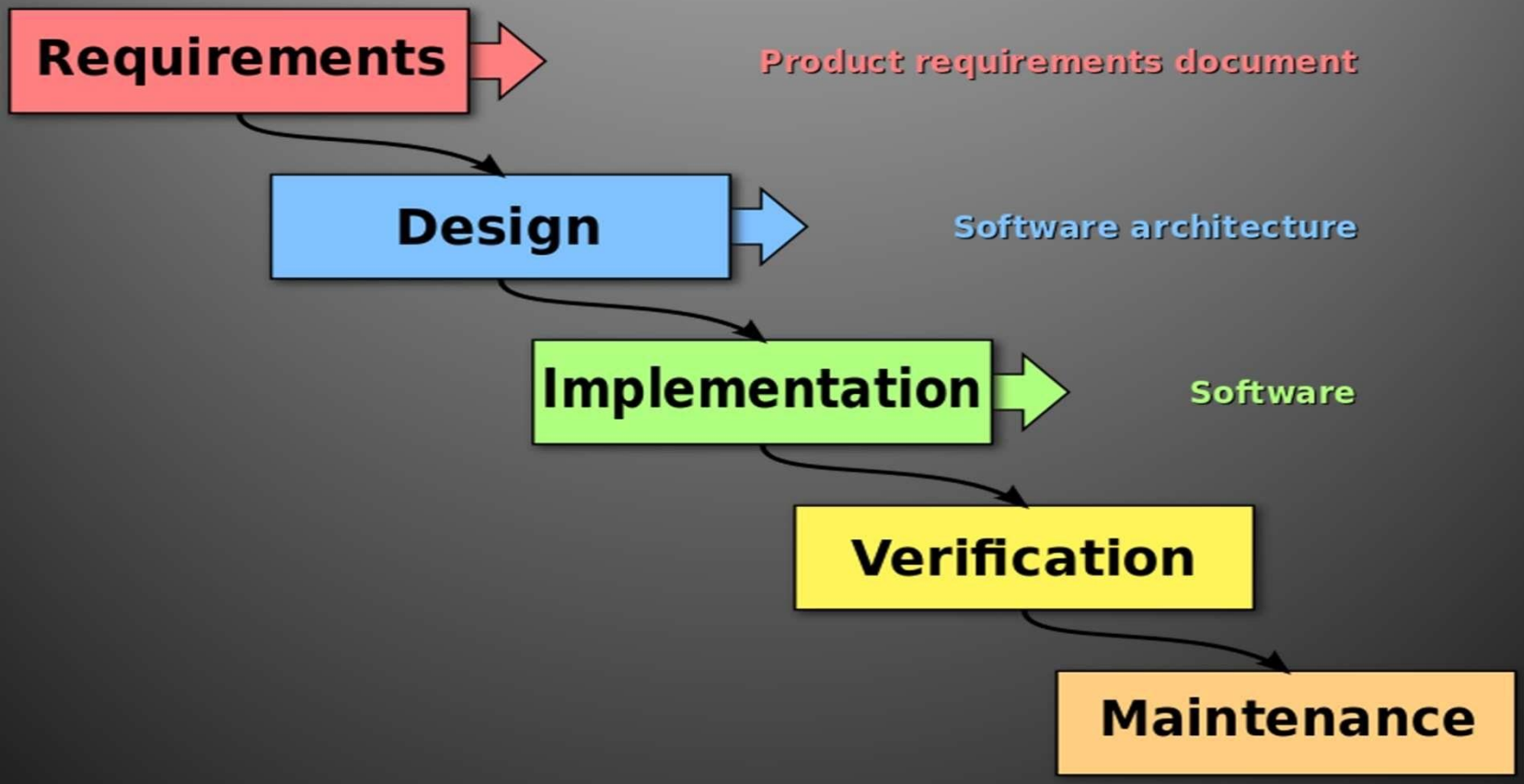
- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.

Software process models

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Reuse-oriented software engineering
  - The system is assembled from existing components. May be plan-driven or agile.

WATERFALL MODEL

This model offers the sequential development process. Because of the cascade from one phase to another this model is called as waterfall model. (Software life cycle)



Waterfall model

- The waterfall model is a sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order.
- In the waterfall model, you must plan and schedule all of the activities before starting working on them (plan-driven process)

Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

The Nature of Waterfall Phases

- In principle, the result of each phase is one or more documents that should be approved and the next phase shouldn't be started until the previous phase has completely been finished.
- In practice, however, these phases overlap and feed information to each other. For example, during design, problems with requirements can be identified, and during coding, some of the design problems can be found, etc.
- The software process therefore is not a simple linear but involves feedback from one phase to another. So, documents produced in each phase may then have to be modified to reflect the changes made.

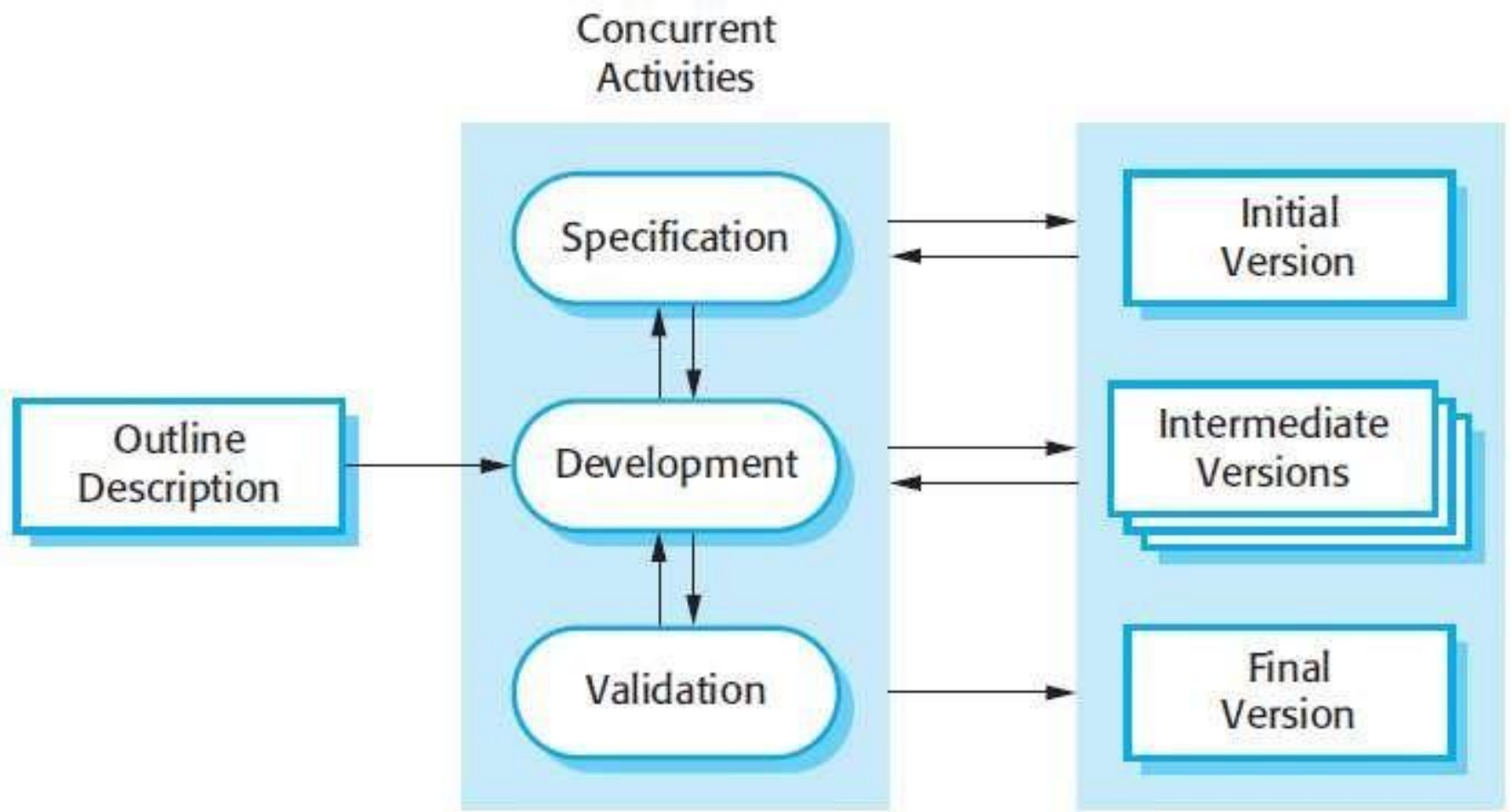
When To Use?

- In principle, the waterfall model should only be applied when requirements are well understood and unlikely to change radically during development as this model has a relatively rigid structure which makes it relatively hard to accommodate change when the process in underway.

Incremental Development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.
- The activities of a process are not separated but interleaved with feedback involved across those activities.

Incremental development



Incremental Development

- Each system increment reflects a piece of the functionality that is needed by the customer. Generally, the early increments of the system should include the most important or most urgently required functionality.
- This means that the customer can evaluate the system at early stage in the development to see if it delivers what's required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

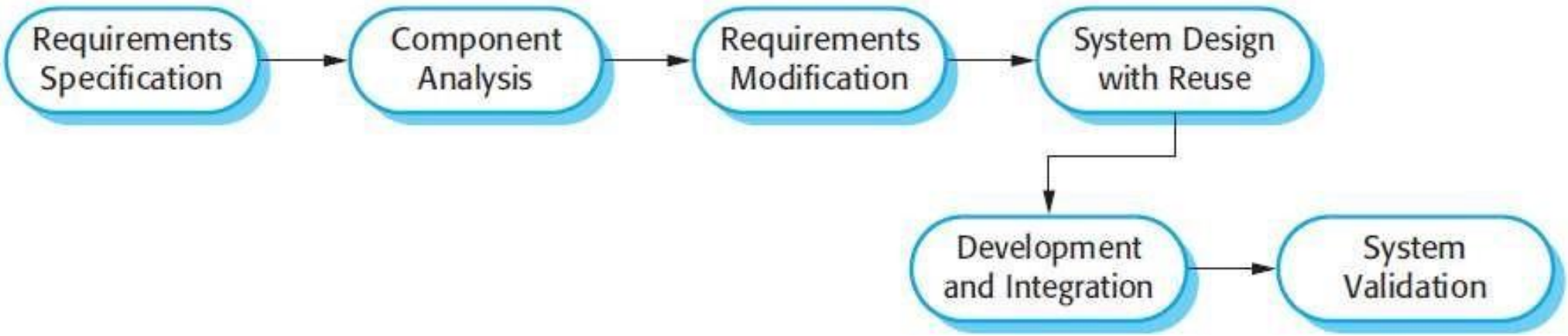
Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
- It is easier to get customer feedback on the development work that has been done.
- More rapid delivery and deployment of useful software to the customer is possible.

Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

Reuse-oriented software engineering



Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system

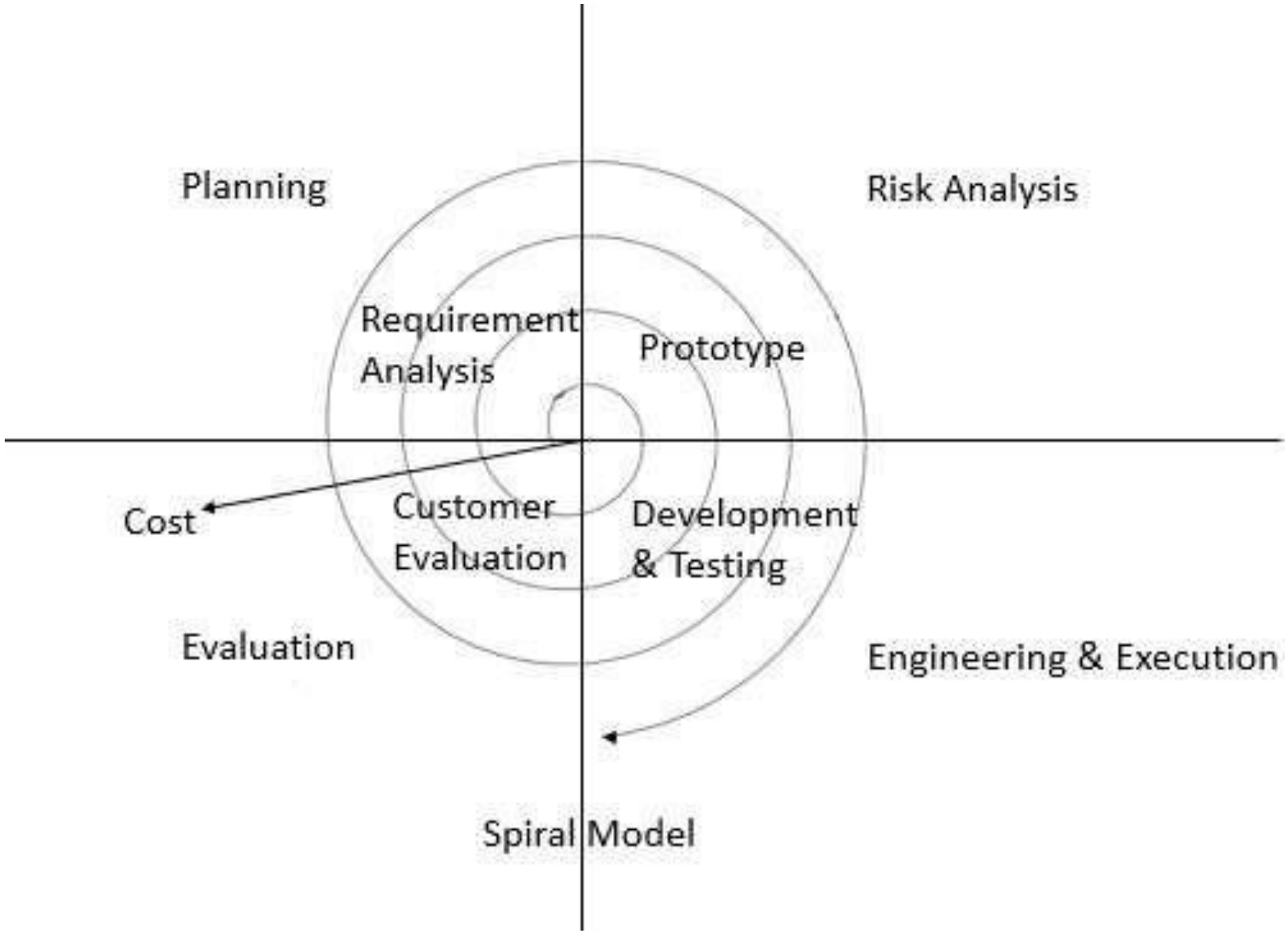
Types of software component

- Web services that are developed according to service standards and which are available for remote invocation.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems that are configured for use in a particular environment.

Spiral model

- The spiral model is similar to the incremental model, with more emphasis placed on risk analysis.
- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
- A software project repeatedly passes through these phases in iterations (called Spirals in this model).

Types of software component



Spiral Model of SDLC spans into four phases

Spiral Model Phases	Activities performed during phase
Planning	• It includes estimating the cost, schedule and resources for the iteration. It also involves understanding the system requirements for continuous communication between the system analyst and the customer
Risk Analysis	• Identification of potential risk is done while risk mitigation strategy is planned and finalized
Engineering	• It includes testing, coding and deploying software at the customer site
Evaluation	• Evaluation of software by the customer. Also, includes identifying and monitoring risks such as schedule slippage and cost overrun

Prototyping

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions.
- So, a prototype is useful when a customer or developer is not sure of the requirements, or of algorithms, efficiency, business rules, response time, etc.

Software Prototype

A prototype can be used in:

In the requirements engineering, a prototype can help with the elicitation and validation of system requirements.

- It allows the users to experiment with the system, and so, refine the requirements. They may get new ideas for requirements, and find areas of strength and weakness in the software.

In the system design, a prototype can help to carry out deign experiments to check the feasibility of a proposed design.

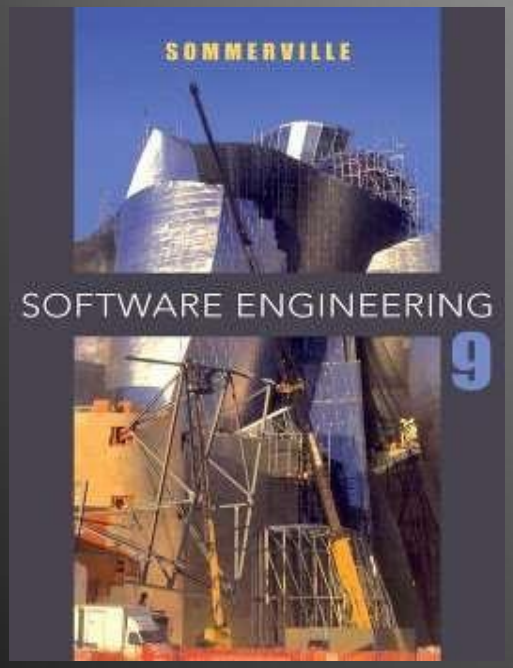
- For example, a database design may be prototype-d and tested to check it supports efficient data access for the most common user queries.

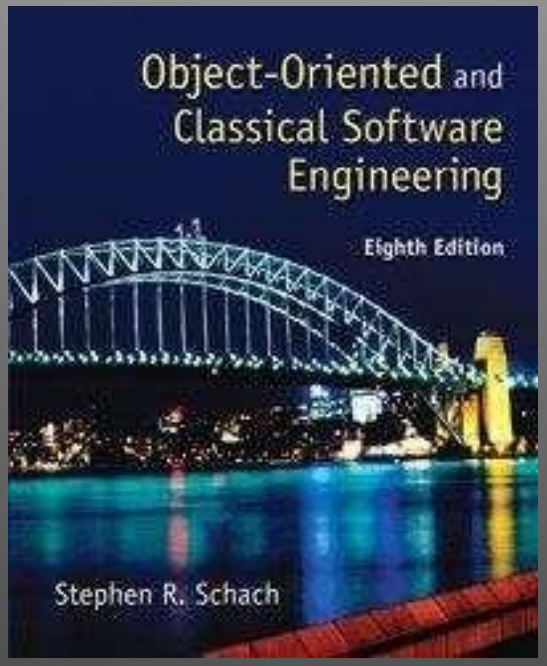
Software Engineering Ethics


S
STRENGTHS
Things your company does well
Qualities that separate you from your competitors
Internal resources such as skilled, knowledgeable staff
Tangible assets such as intellectual property, capital, proprietary technologies etc.
W
WEAKNESSES
Things your company lacks
Things your competitors do better than you
Resource limitations
Unclear unique selling proposition
O
OPPORTUNITIES
Underserved markets for specific products
Few competitors in your area
Emerging need for your products or services
Press/media coverage of your company
T
THREATS
Emerging competitors
Changing regulatory environment
Negative press/ media coverage
Changing customer attitudes toward your company
WordStream

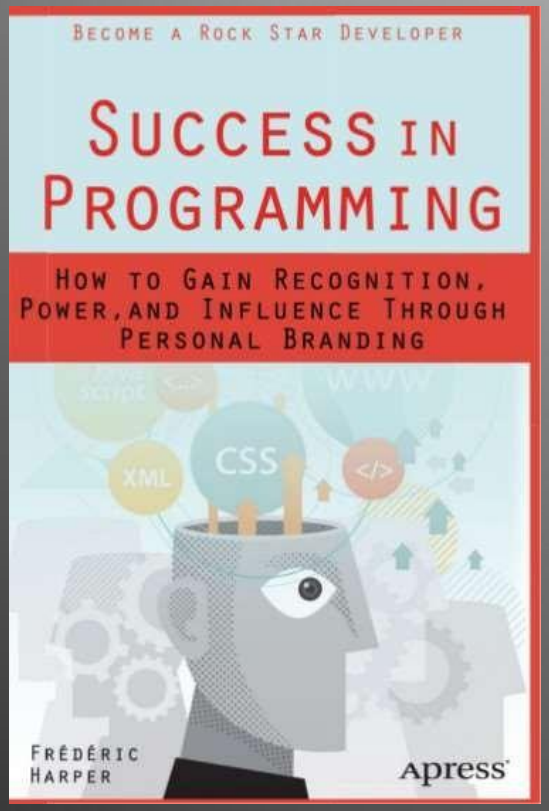
Thank you!

Reference

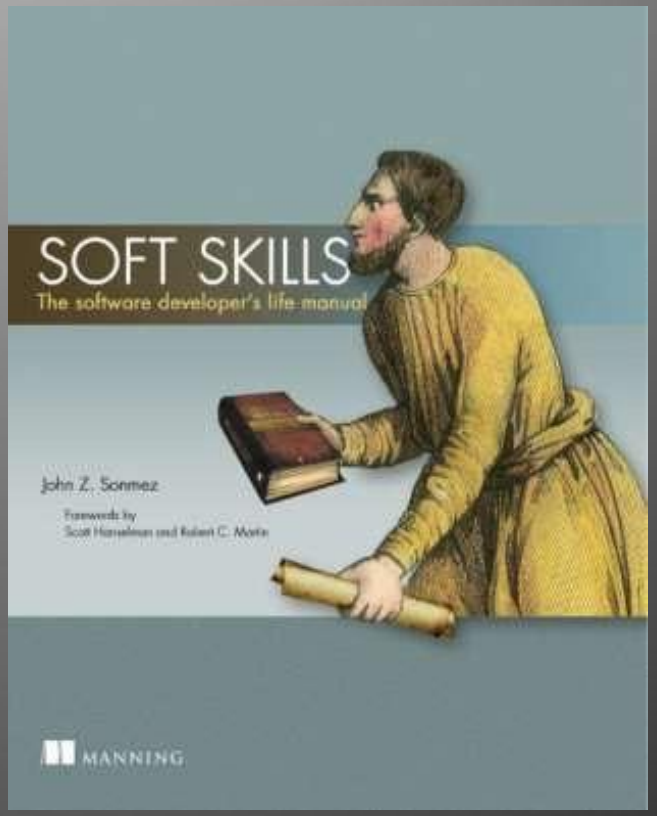

SOMMERVILLE
SOFTWARE ENGINEERING
9


Object-Oriented and Classical Software Engineering
Eighth Edition
Stephen R. Schach


WHAT EVERY ENGINEER SHOULD KNOW ABOUT
SOFTWARE ENGINEERING
Phillip A. Laplante



Success in Programming



SOFT SKILLS

The developer's life manual

Beshoy adel demian

01285687100
beshoy.adel.demian@gmail.com
beshoyadel@iti.gov.eg

https://www.linkedin.com/in/beshoy
adeldemian/