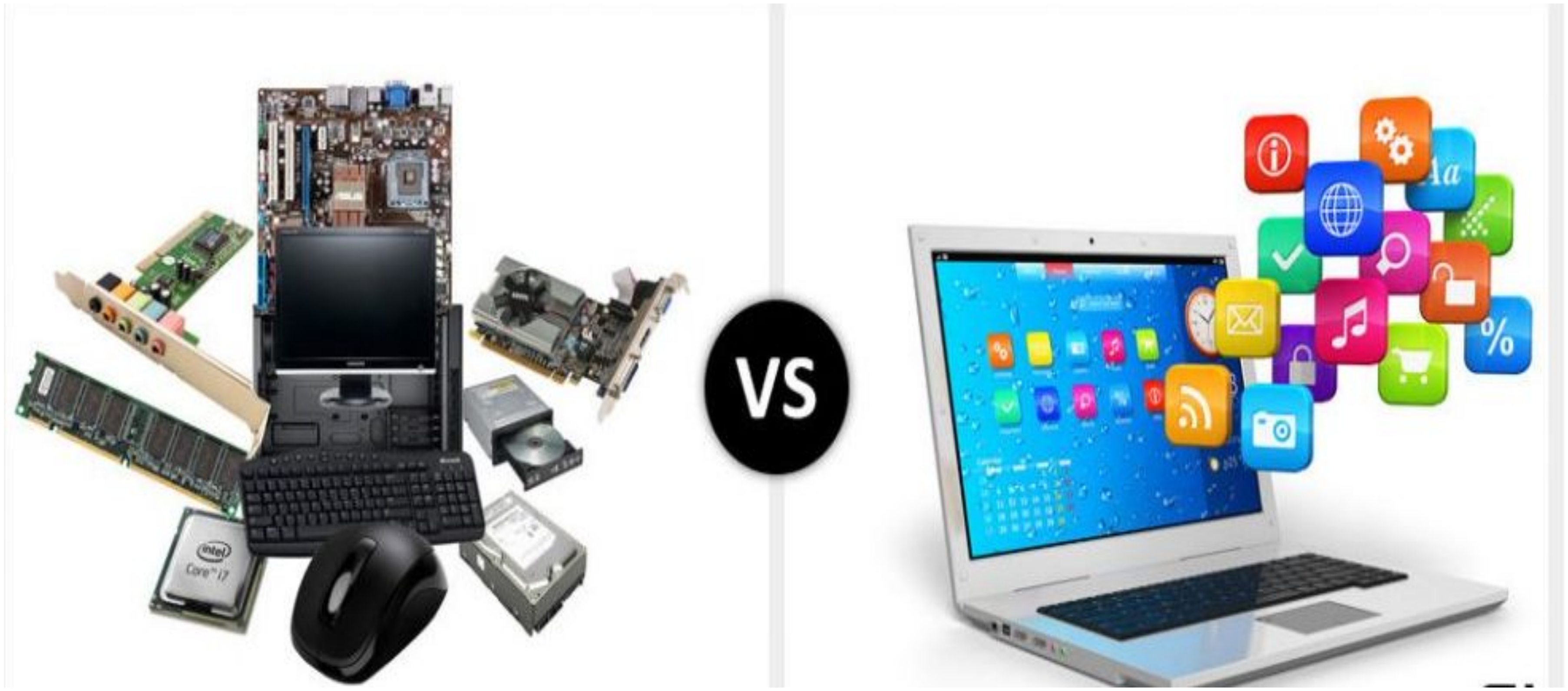
Introduction

Computers are important in people's everyday lives.

Hardware vs Software



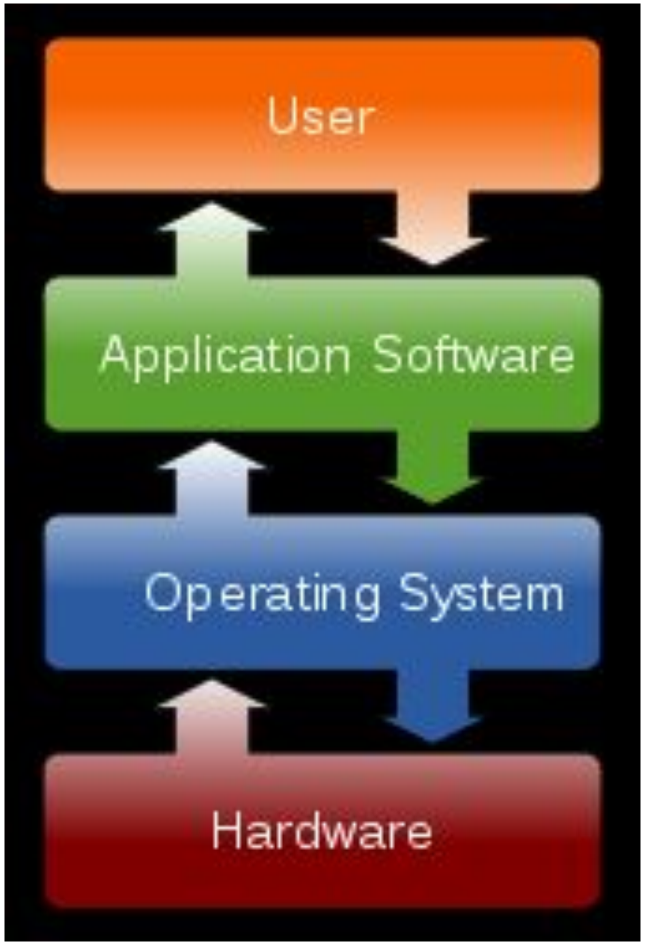
Physical Components of our Computer

Computer Program that do the work

How does Software work?

- Algorithms have to be written.
- Algorithm turned into computer language.
- Another specialized program turned this program into something that computer understands

Changing Nature of Software



Changing Nature of Software

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product line software
- Web applications
- Artificial intelligence software

Evolving Role of Software

Software is a product

•Transforms information - produces, manages, acquires, modifies, displays, or transmits information

•Delivers computing potential of hardware and networks

Software is a vehicle for delivering a product

•Controls other programs (operating system)

•Effects communications (networking software)

•Helps build other software (software tools & environments)

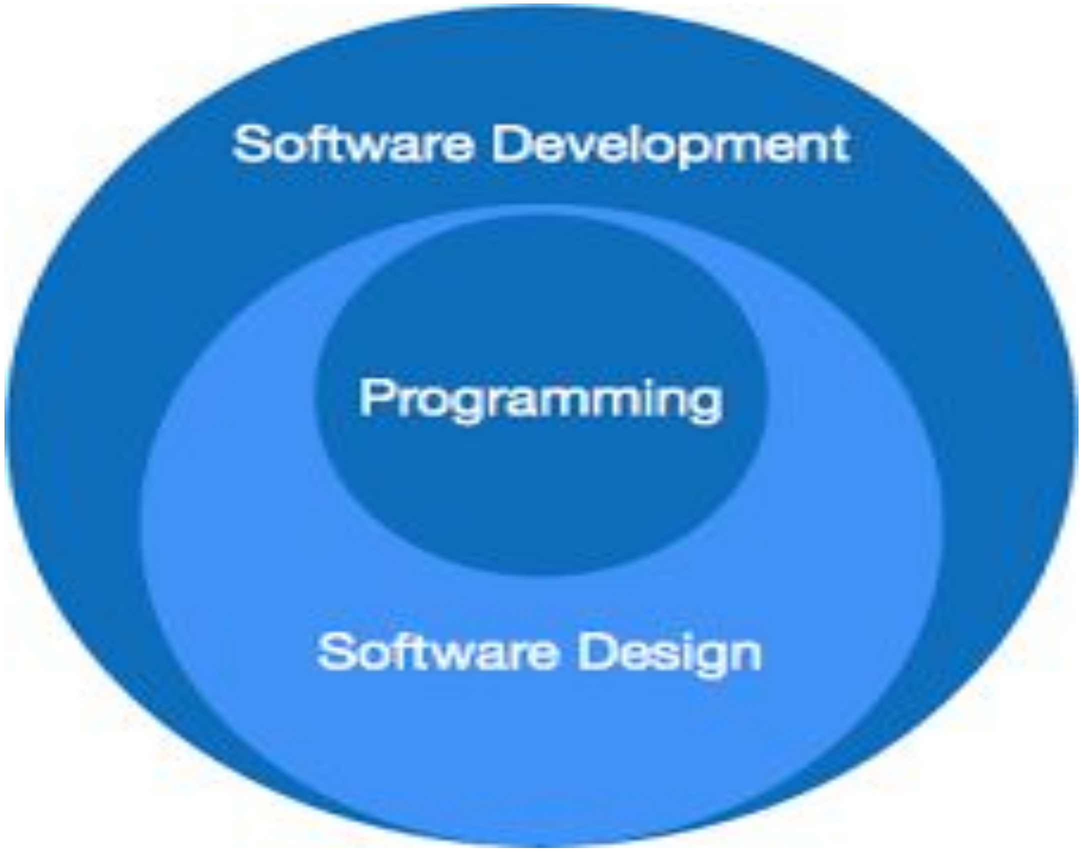
Software Characteristics

Software is developed or engineered; it is not manufactured.

Software does not “wear out” but it does deteriorate.

Software continues to be custom built, as industry is moving toward component based construction.

Software Development

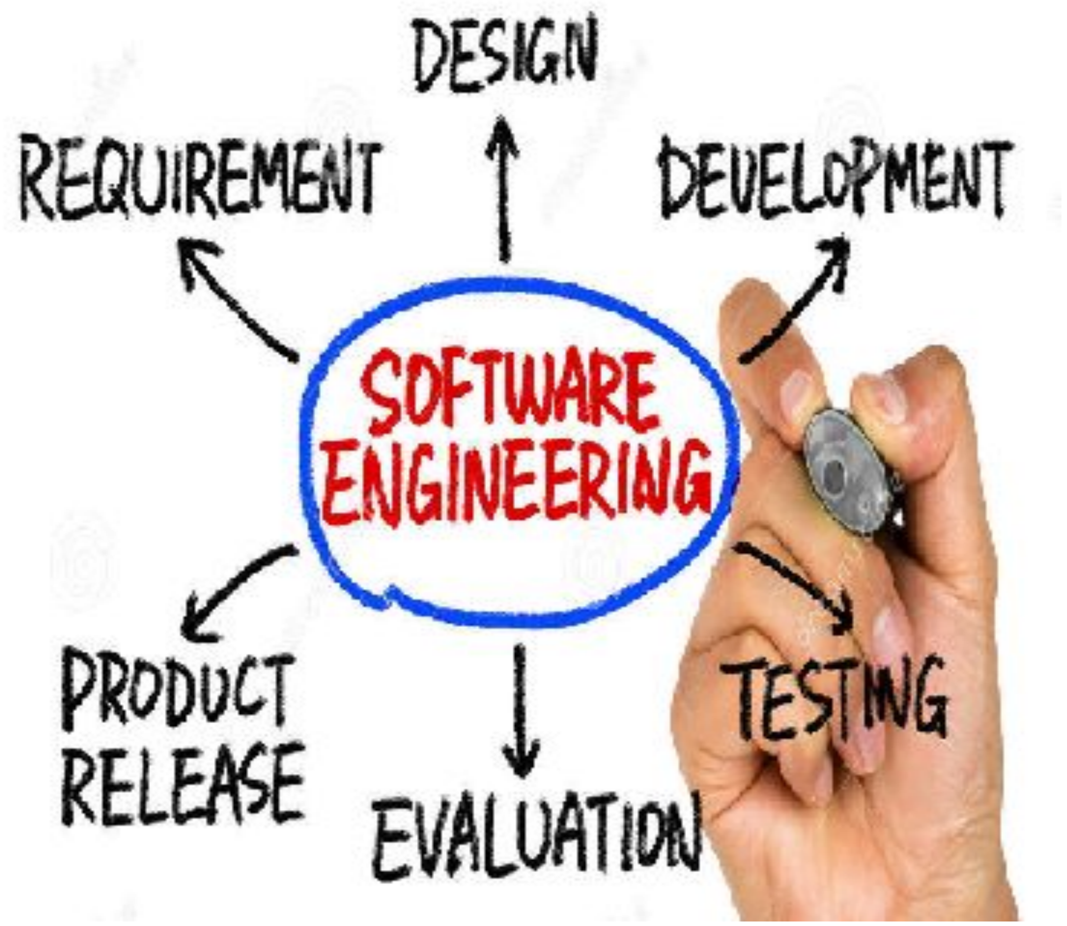


Software Engineering

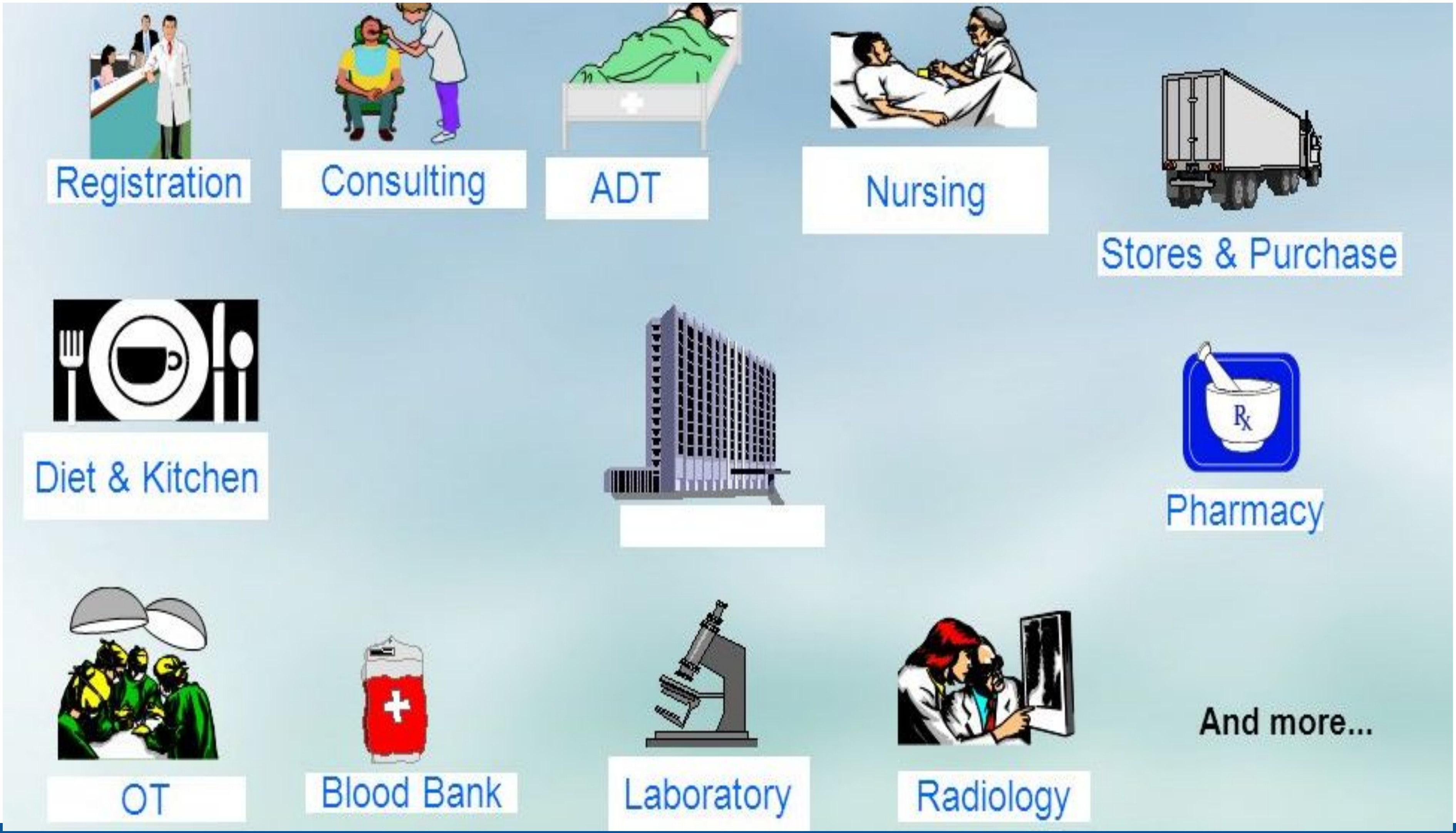
- **Software engineering** is a field of engineering, for designing and writing programs for computers or other electronic devices.

- **Engineering** forces us to focuses on systematic, scientific and well defined process to produce a good quality product.

Software Engineering



Software Modules for Hospital ERP Software



Software Modules for University Management Software

Software Application Domains

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product line software
- Web applications
- Artificial intelligence software
- Open World Computing
- Netsourcing
- Open Source

Legacy Software

Older software that is still useful.

Typical problems with legacy systems:

- Original developers not available
- Outdated development methods used
- Extensive patches and modifications have been made
- Missing or outdated documentation

Software Crisis

- **Software crisis** is a term used in the early days of computing science for the difficulty of writing useful and efficient computer programs in the required time.

- The major causes of software crisis are the problems associated with poor quality software such as
- Malfunctioning of software systems
- Inefficient development of software
- Dissatisfaction amongst the users of the software.

Software Crisis – Gulf War

During the Gulf War in 1991, the United States of America used Patriot missiles as a defence against Iraqi Scud missiles. However, the Patriot failed to hit the Scud many times. As a result, 28 US soldiers were killed in Dhahran, Saudi Arabia. An inquiry into the incident concluded that a small bug had resulted in the miscalculation of missile path.

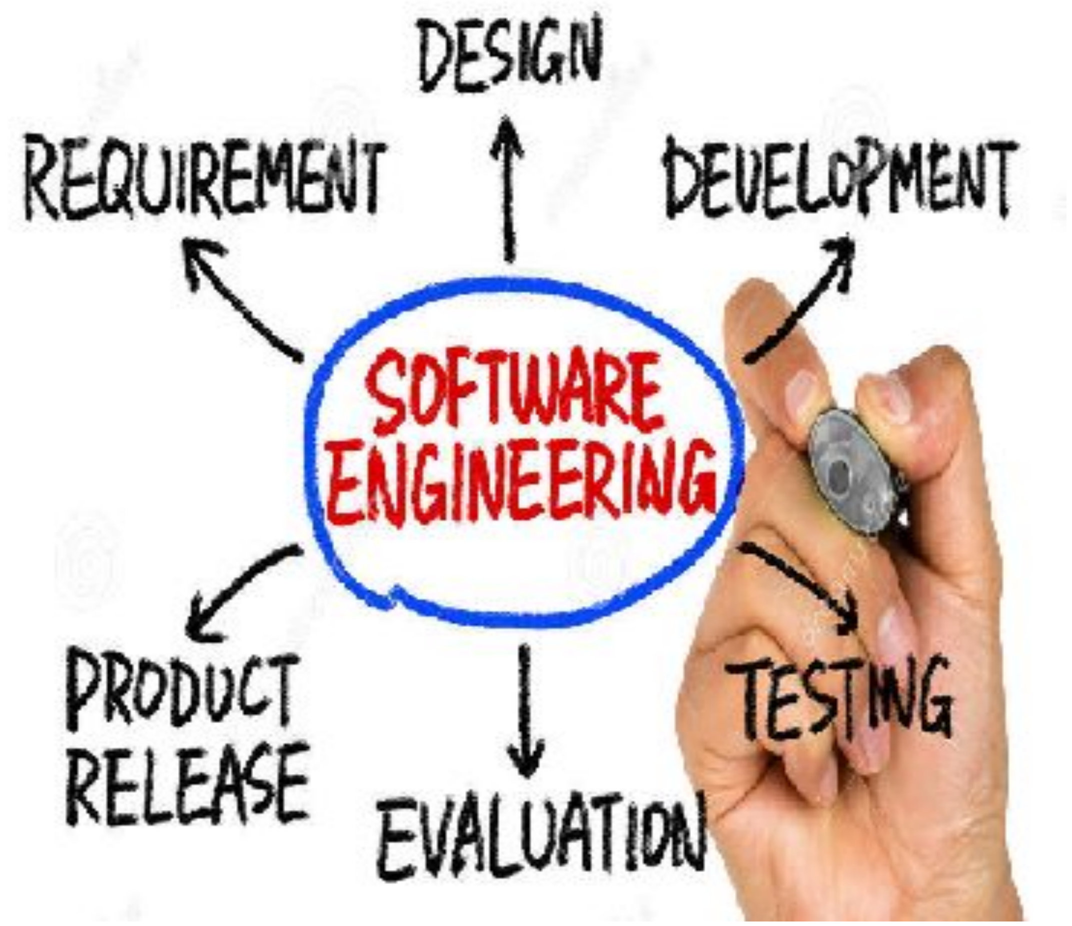
Software Crisis – Rocket Problem

In 1996, Arian-5 space rocket, developed at the cost of $7000 million over a period of 10 years was destroyed within less than a minute after its launch. The crash occurred because there was a software bug in the rocket guidance system.

Software Crisis – Y2K Problem

Year 2000 (Y2K) problem refers to the widespread snags in processing dates after the year 2000. The roots ofY2K problem can be traced back to 1960-80 when developers shortened the 4-digit date format like 1972 to a 2-digit format like 72 because of limited memory. At that time they did not realize that year 2000 will be shortened to 00 which is less than 72. In the 1990s, experts began to realize this major shortcoming in the computer application and then millions were spent to handle this problem.

Software Engineering



Participants

Customer

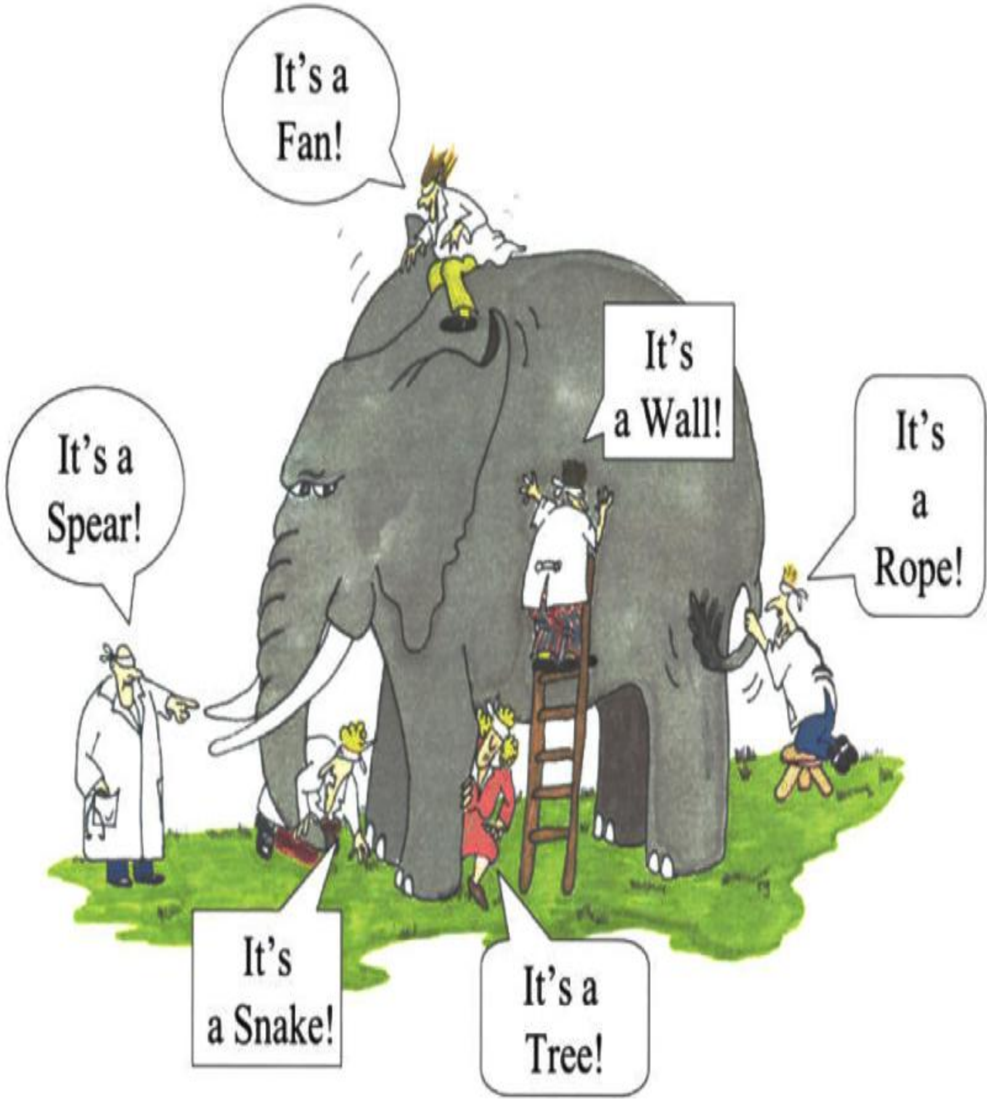
Manager

Developer

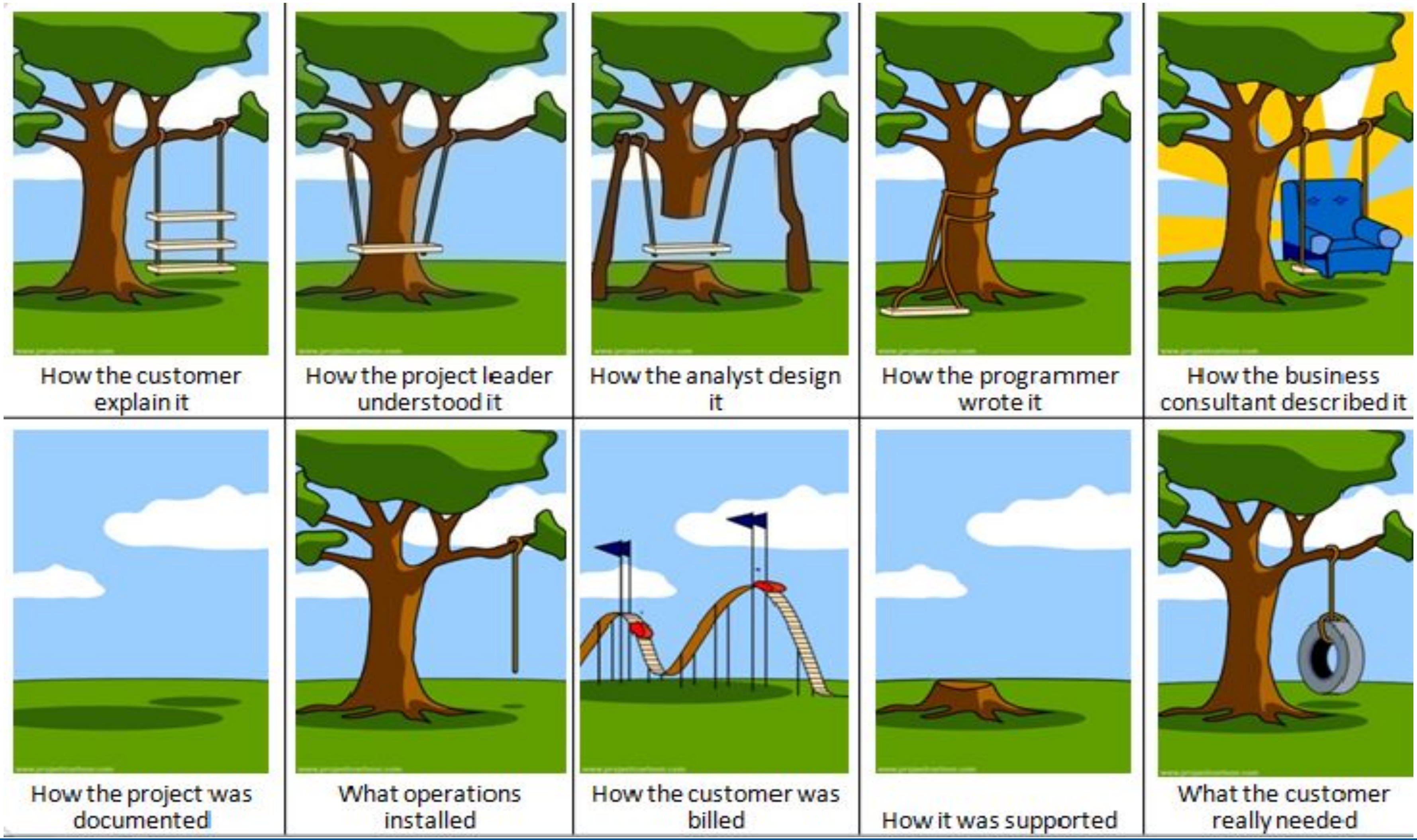
Software Myths

Misleading attitudes that have caused serious problems.

A number of common beliefs or myths that software managers, Customers and Practitioners (developers) belief falsely.



Software Myths


How the customer explain it
How the project leader understood it
How the analyst design it
How the programmer wrote it
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Software Myths – Management

"We already have a book that is full of standards and procedures for building software. Won't that provide my people with everything they need to know?"

Not used, not up to date, not complete, not focused on quality, time, and money

"If we get behind, we can add more programmers and catch up"

Adding people to a late software project makes it later Training time, increased communication lines

"If I decide to outsource the software project to a third party, I can just relax and let that firm build it"

Software projects need to be controlled and managed

Software Myths – Customer

"A general statement of objectives is sufficient to begin writing programs – we can fill in the details later"

Ambiguous statement of objectives spells disaster

"Project requirements continually change, but change can be easily accommodated because software is flexible"

Impact of change depends on where and when it occurs in the software life cycle (requirements analysis, design, code, test)

Software Myths – Practitioner’s

"Once we write the program and get it to work, our job is done"

60% to 80% of all effort expended on software occurs after it is delivered

"Until I get the program running, I have no way of assessing its quality

Formal technical reviews of requirements analysis documents, design documents, and source code (more effective than actual testing)

"The only deliverable work product for a successful project is the working program"

Software, documentation, test drivers, test results

"Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down"

Creates quality, not documents; quality reduces rework and provides software on time and within the budget

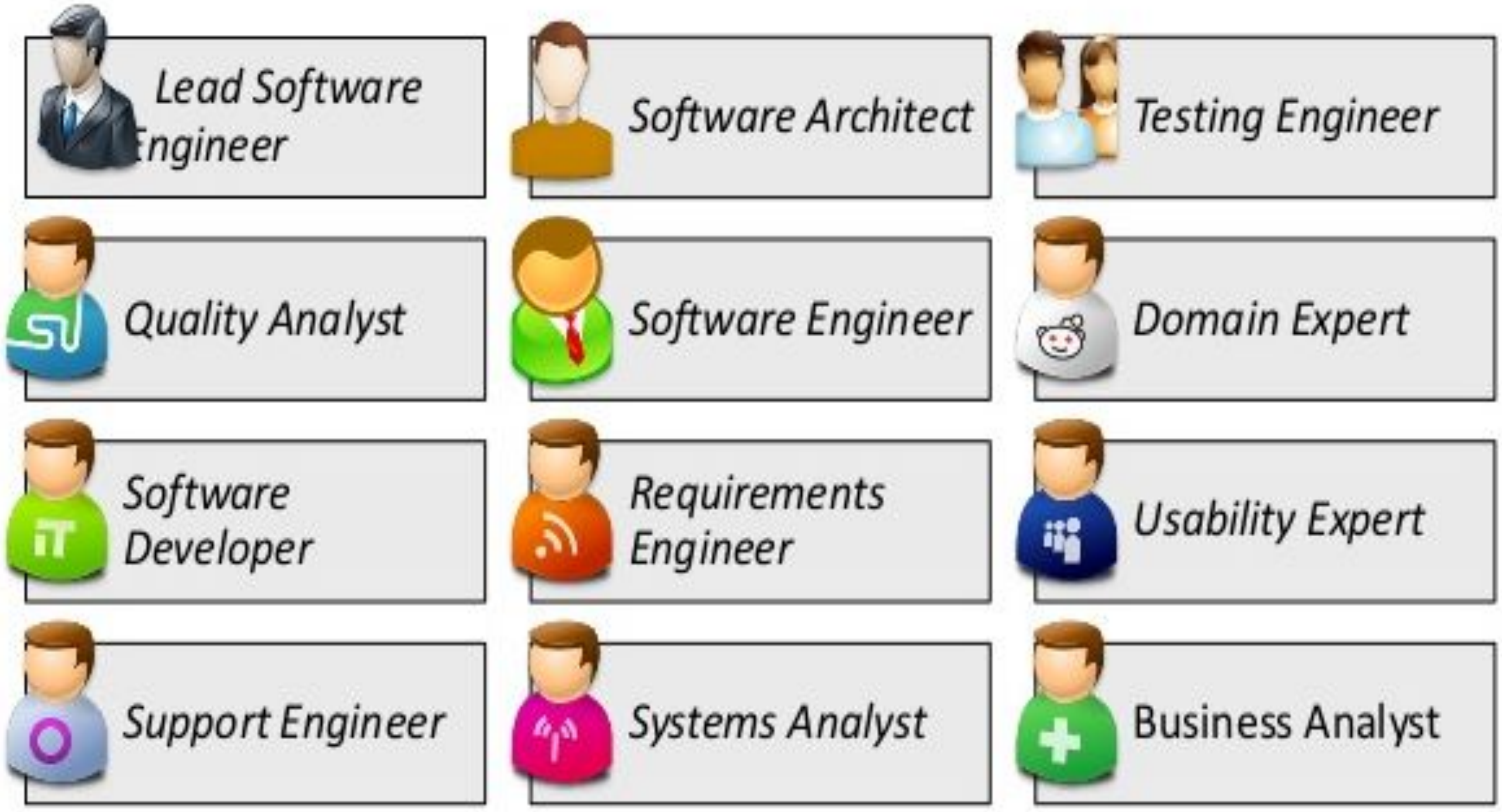
Software Engineering Practices

- **Understand the problem**
  - Communication & Analysis
- **Plan the solution**
  - Modelling and Software Design
- **Carryout the plan**
  - Code Generation
- **Examine the result for accuracy**
  - Testing and Quality Assurance

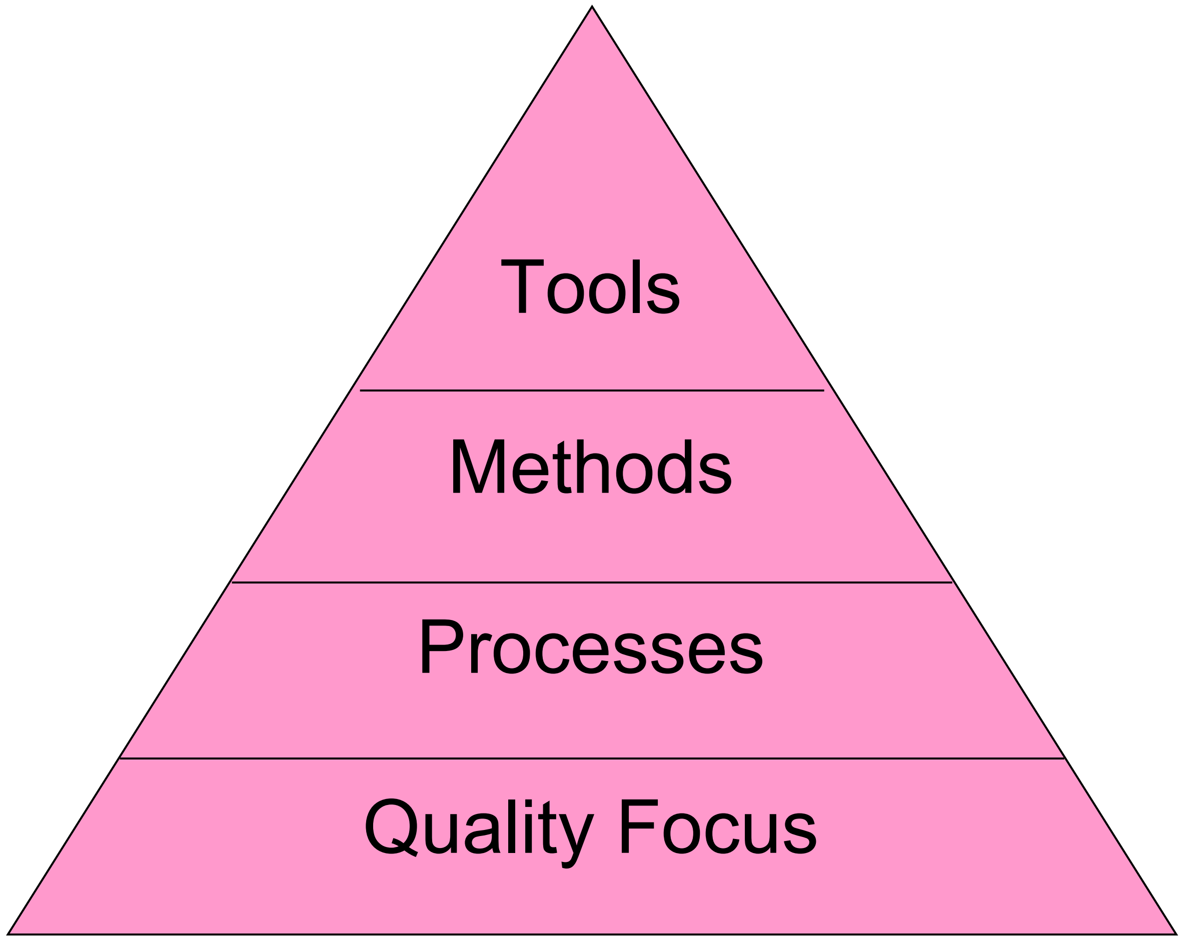
Software Engineering

DESIGN
REQUIREMENT
DEVELOPMENT
SOFTWARE ENGINEERING
PRODUCT RELEASE
TESTING
EVALUATION

Software Professionals



Software Engineering Layered Technology



Software Process

Software Process

A **framework** for the activities, actions, and tasks that are required to build high-quality software.

Activity – strives to achieve a broad objective
Action – encompasses a set of tasks that produce a major work product.
Task – well defined objective that produces a tangible outcome

Social Learning Process

Software is embodied **knowledge that is initially dispersed, tacit and incomplete.**

In order to **convert knowledge into software**, dialogues are needed between users and designers, between designers and tools to bring knowledge into software.

Software development is essentially an iterative social learning process, and the **outcome is "software capital".**

Software Engineering General Principles

- The reason all it exists
- KISS (Keep It Simple, Stupid!)
- Maintain the Vision
- What You Produce, Others will Consume
- Be Open to the Future
- Plan ahead for Reuse
- Think…..!

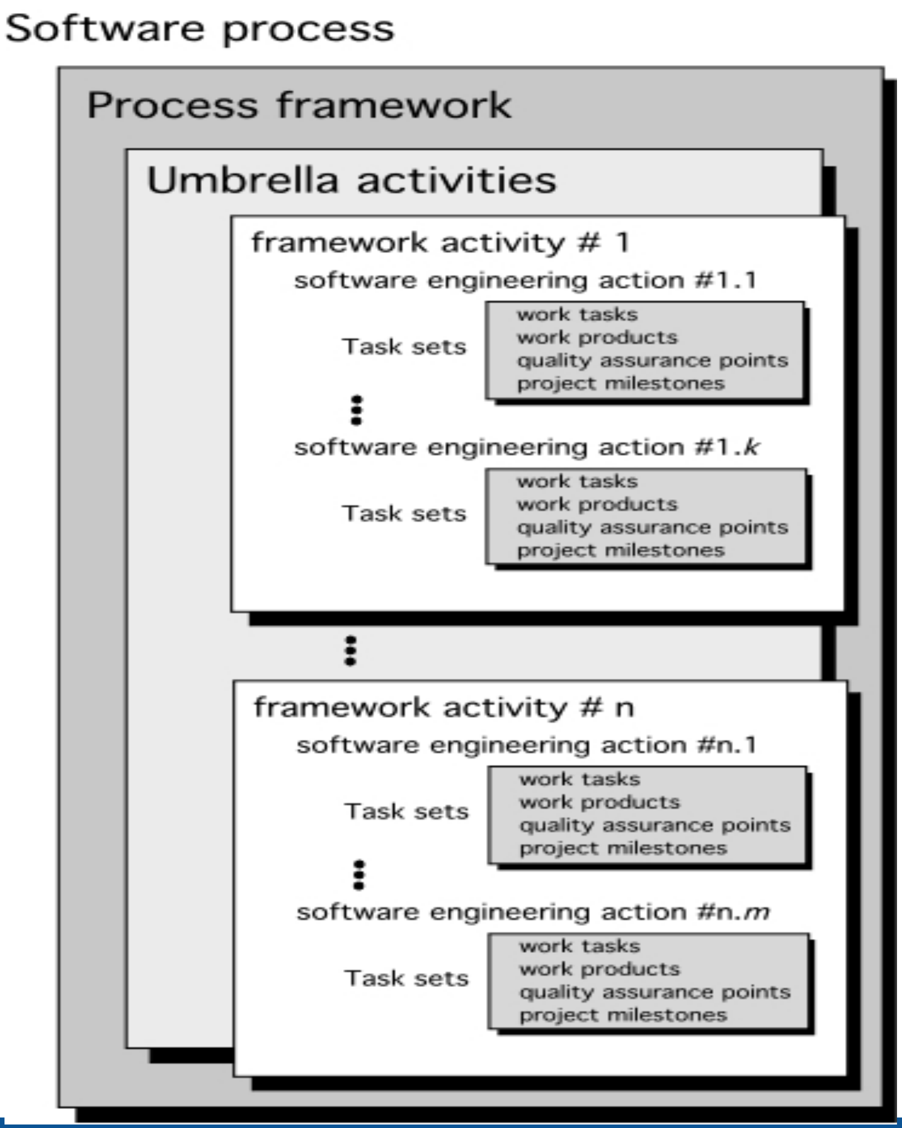
Generic Process Framework Activities

- Communication
- Planning
- Modelling
- Construction
- Deployment

Umbrella Activities

- Software project tracking and control
- Risk Management
- Software Quality Assurance
- Technical Reviews
- Measurement
- Software Configuration Management
- Reusability Management
- Work Product preparation and production

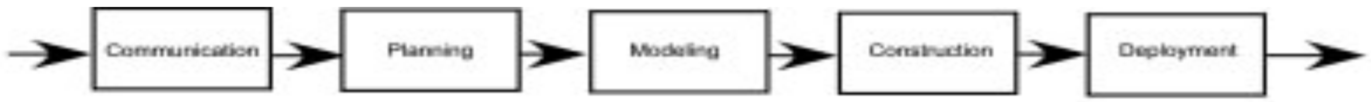
Process Framework



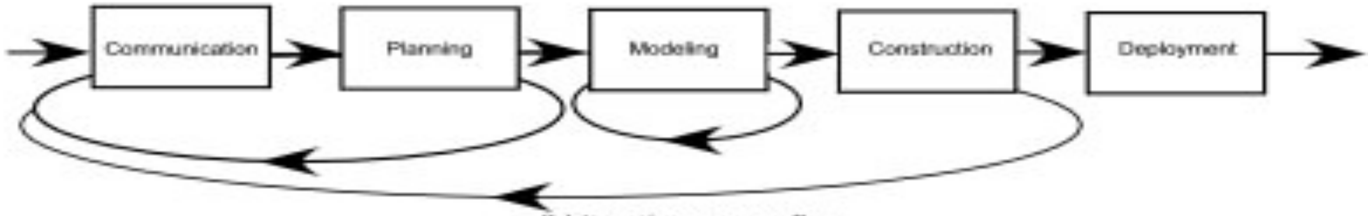
Process Flow

How the framework activities and the actions are organized with respect to sequence and time

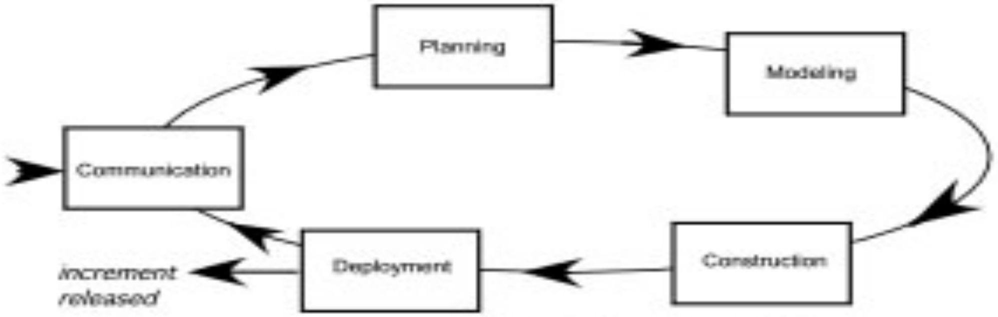
- **Linear Process Flow**
- **Iterative Process Flow**
- **Evolutionary Process Flow**
- **Parallel Process Flow**



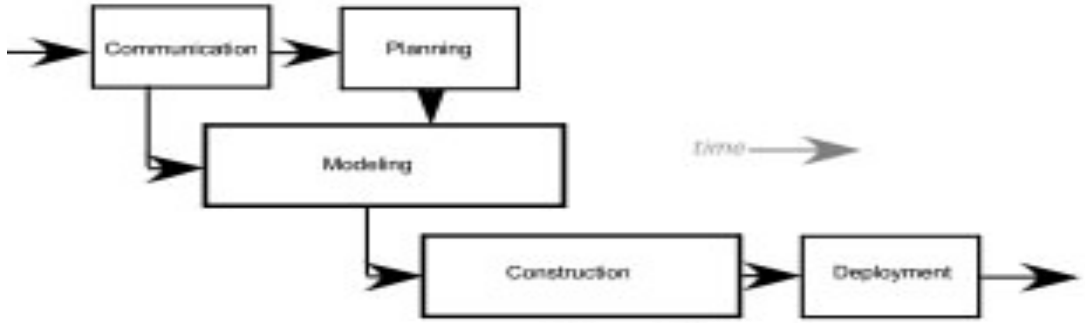
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

Before you can proceed with Process Model…

What **actions** are appropriate for a framework activity given the nature of the problem, the characteristics of the people and the stakeholders?

- A list of the task to be accomplished
- A list of the work products to be produced
- A list of the quality assurance filters to be applied

Identifying a Task Set

Defines the actual work to be done to accomplish the objectives of software engineering action.

- Depending on the need and size of the Software Project
- Collection of software engineering wok tasks

Requirement Gathering Example

A small software project requested by one person with simple requirements. The work tasks of this action are:

1. Make contact with stakeholder via telephone.
2. Discuss requirements and take notes.
3. Organize notes into a brief written statement of requirements.
4. E-mail to stakeholder for review and approval.

Requirement Gathering Example

- Make a list of stakeholders for the project.
- Invite all stakeholders to an informal meeting.
- Ask each stakeholder to make a list of features and functions required.
- Discuss requirements and build a final list.
- Prioritize requirements.
- Note areas of uncertainty.

Process Pattern

- Process-related problem that is encountered during software engineering work
- Identifies the environment in which the problem has been encountered
- Suggests one or more proven solutions to the problem.

Process Pattern Types

Stage patterns—defines a problem associated with a framework activity for the process. It includes multiple task patterns as well. Eg., **EstablishingCommunication** would incorporate the task pattern **RequirementsGathering** and others.

Task patterns—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice.

Phase patterns—defines the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.

Process Pattern - Abstraction

1. Problems and solutions associated with a complete process model (e.g. prototyping).

2. Problems and solutions associated with a framework activity (e.g. planning) or

3. An action with a framework activity (e.g. project estimating).

Example for Process Pattern

Describes an approach that may be applicable when stakeholders have a general idea of what must be done but are unsure of specific software requirements. *(Phase Pattern)*

Pattern name: Requirements Unclear

Intent: This pattern describes an approach for building a model that can be assessed iteratively by stakeholders in an effort to identify or solidify software requirements.

Initial context. Conditions must be met (1) stakeholders have been identified; (2) a mode of communication between stakeholders and the software team has been established; (3) the overriding software problem to be solved has been identified by stakeholders ; (4) an initial understanding of project scope, basic business requirements and project constraints has been developed.

Problem. Requirements are hazy or nonexistent. stakeholders are unsure of what they want.

Solution. A description of the prototyping process would be presented here.

Resulting context. A software prototype that identifies basic requirements. (modes of interaction, computational features, processing functions) is approved by stakeholders. Following this, 1. This prototype may evolve through a series of increments to become the production software or 2. the prototype may be discarded.

Process Assessment and Improvement

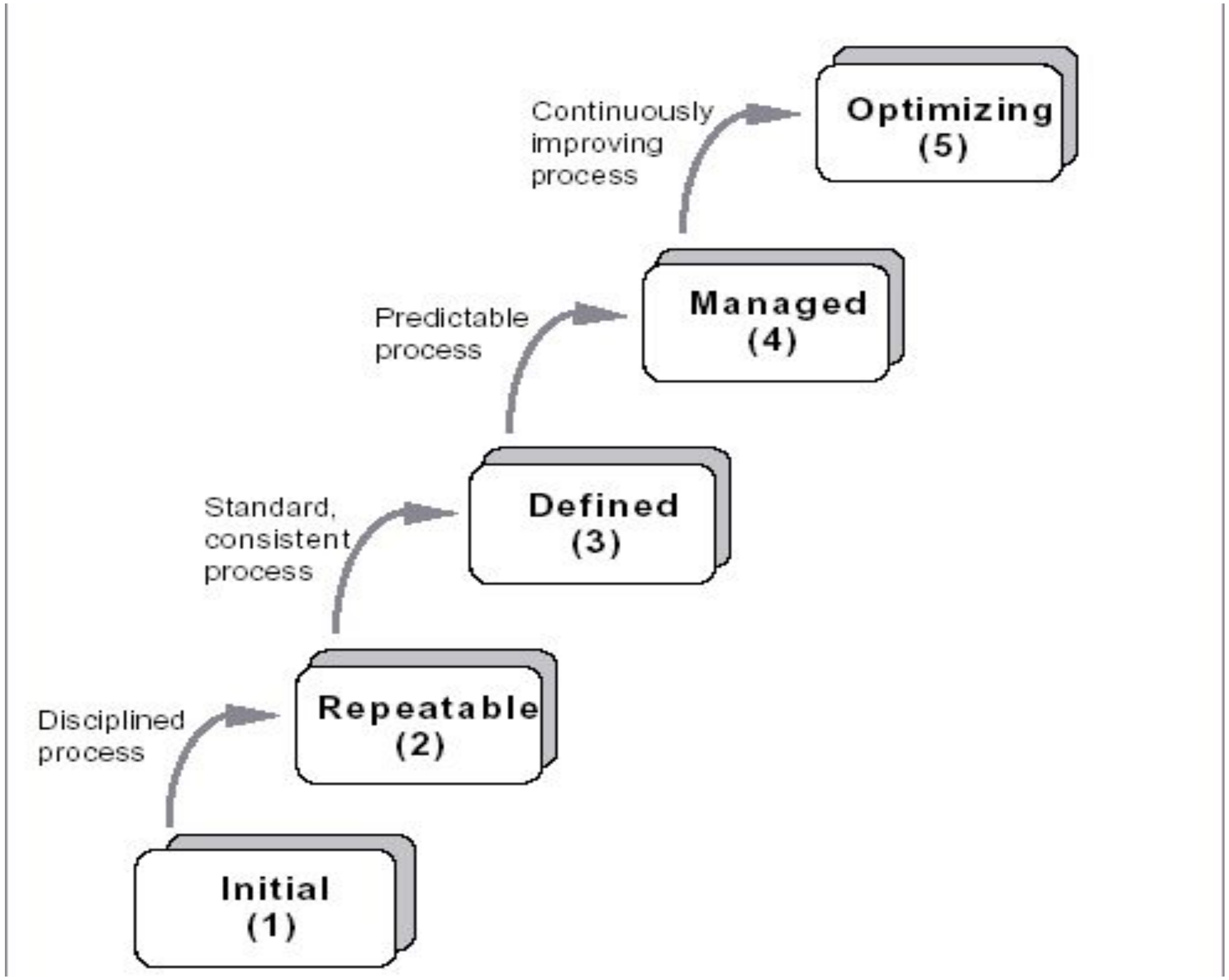
Standard CMMI Assessment Method for Process Improvement (SCAMPI) — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.

CMM-Based Appraisal for Internal Process Improvement (CBA IPI)—provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment.

SPICE—The SPICE (ISO/IEC15504) standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process.

ISO 9001:2000 for Software— a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies.

Five Levels of Process Maturity



If a software production gets behind schedule, one can add more programmers and catch up.

a) True
b) False

The only deliverable work product for a successful project is the working program.

a) True
b) False

A general statement of objectives is the major cause of failed software efforts.

a) True
b) False

Arrange the following steps to form a basic/general Engineering Process Model.

i. Test
ii. Design
iii. Install
iv. Specification
v. Manufacture
vi. Maintain

"Software engineers should not use their technical skills to *misuse* other people's computers."Here the term *misuse* refers to:

a) Unauthorized access to computer material
b) Unauthorized modification of computer material
c) Dissemination of viruses or other malware
d) All of the mentioned

Efficiency in a software product does not include ________

a) responsiveness
b) licensing
c) memory utilization
d) processing time

Company has latest computers and state-of the- art software tools, so we shouldn’t worry about the quality of the product.

a) True
b) False

______________&______________ are two kinds of software products.

a) CAD, CAM

b) Firmware, Embedded

c) Generic, Customised

Which one of the following is not an Umbrella Activity that complements the five process framework activities and help team manage and control progress, quality, change, and risk.

a) Re-usability management
b) Risk management
c) Measurement
d) User Reviews
e) Software quality assurance

Prescriptive Process Models

Originally proposed to bring order to the chaos of software development.

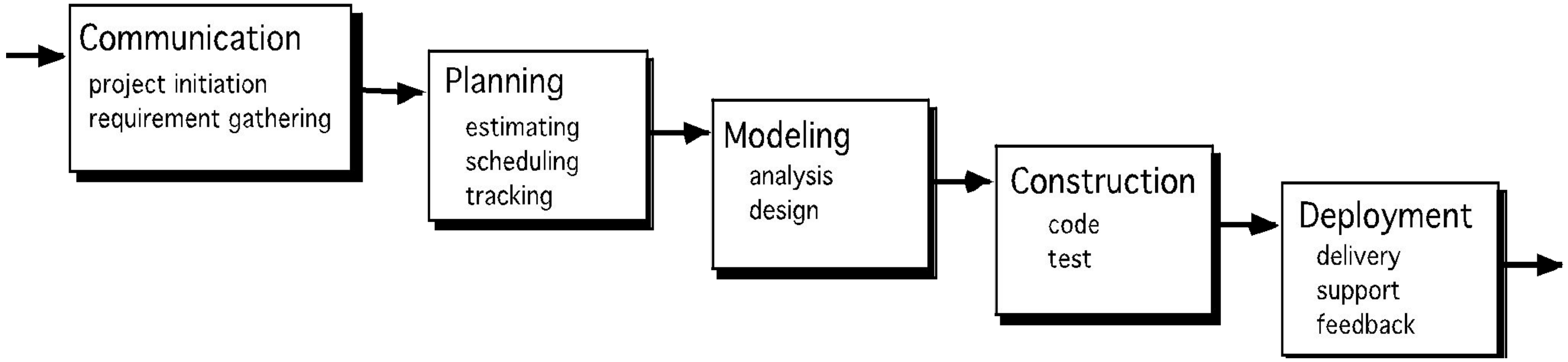
Defines a prescribed set of process elements and a predictable process wok flow.

Prescriptive Process Model

- The Waterfall Model
- Incremental Process Model
- Evolutionary Process Model
  - Prototyping
  - Spiral Model
- Concurrent Model

The Waterfall Model

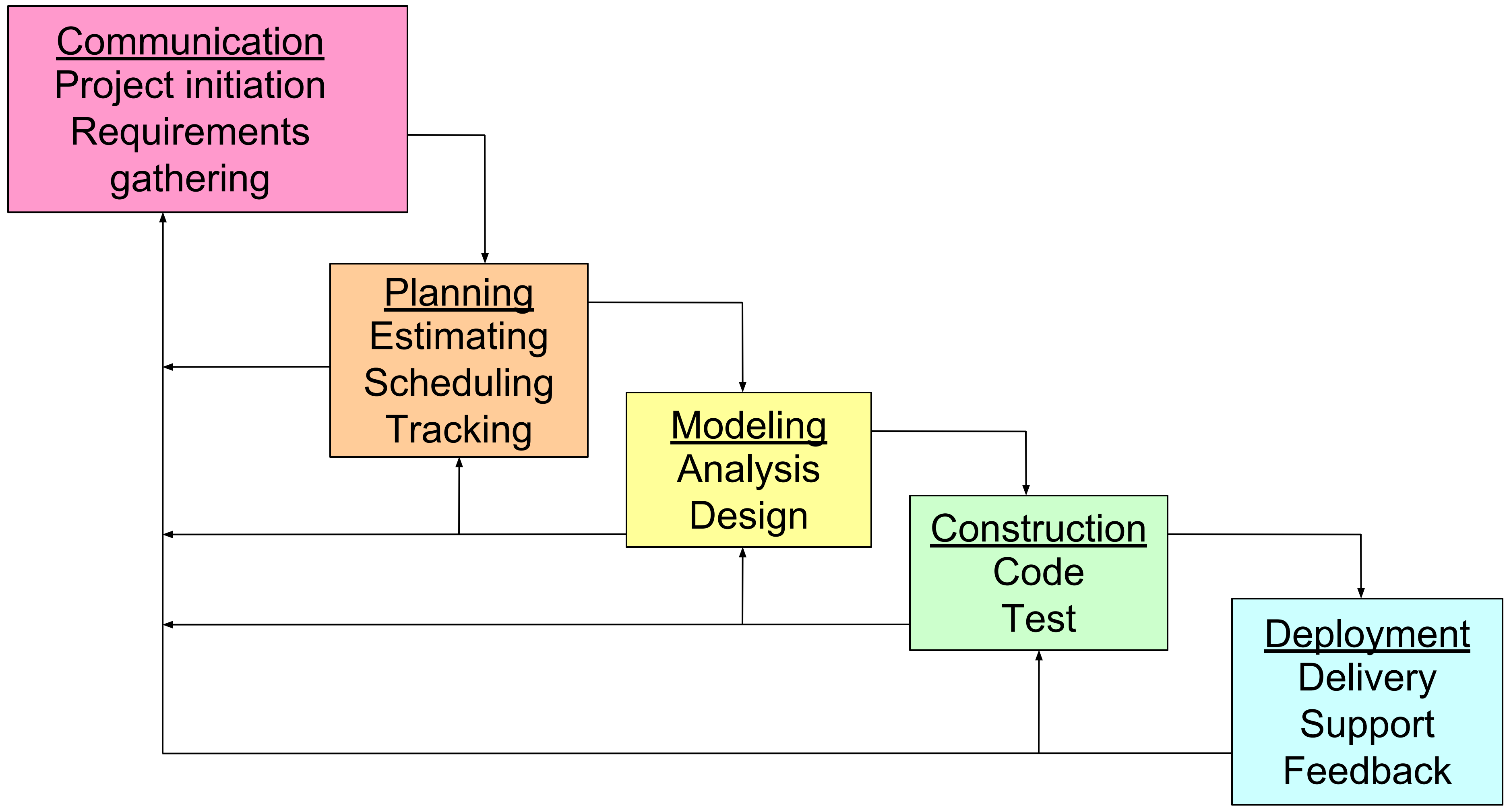
The classic life cycle suggests a systematic, sequential approach to software development.



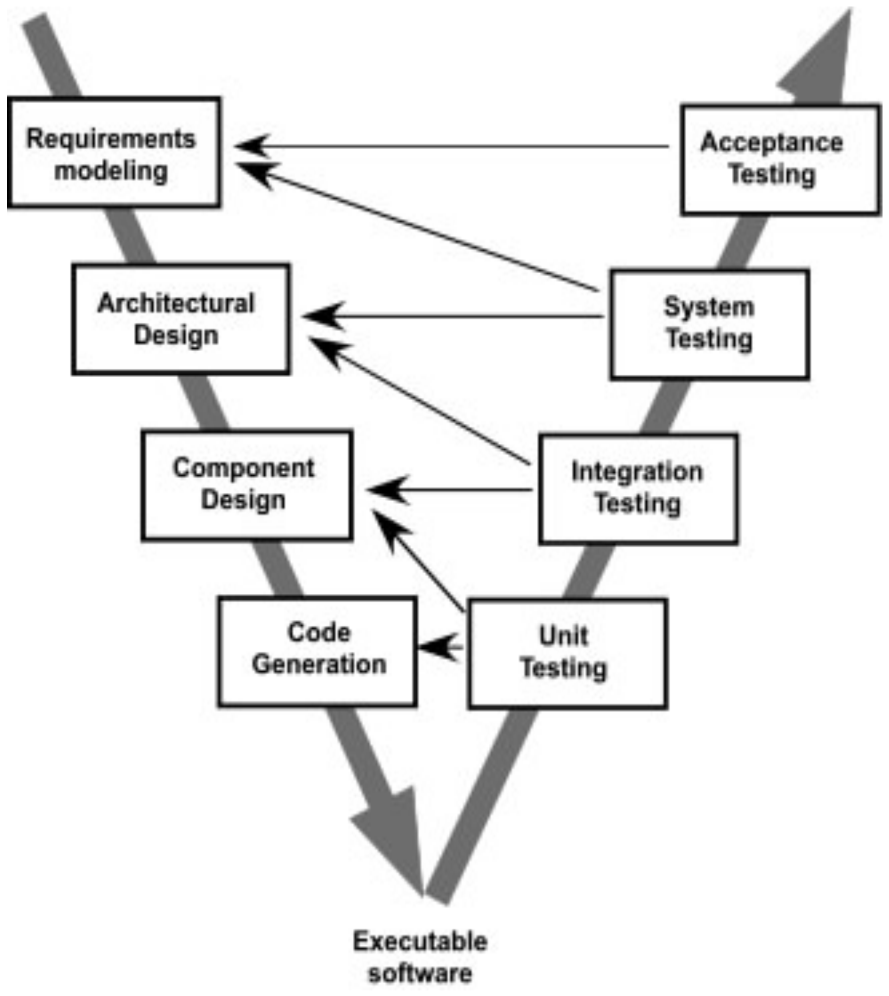
Problems:

1. Rarely linear, iteration needed.
2. Hard to state all requirements explicitly
3. Code will not be released until completion.

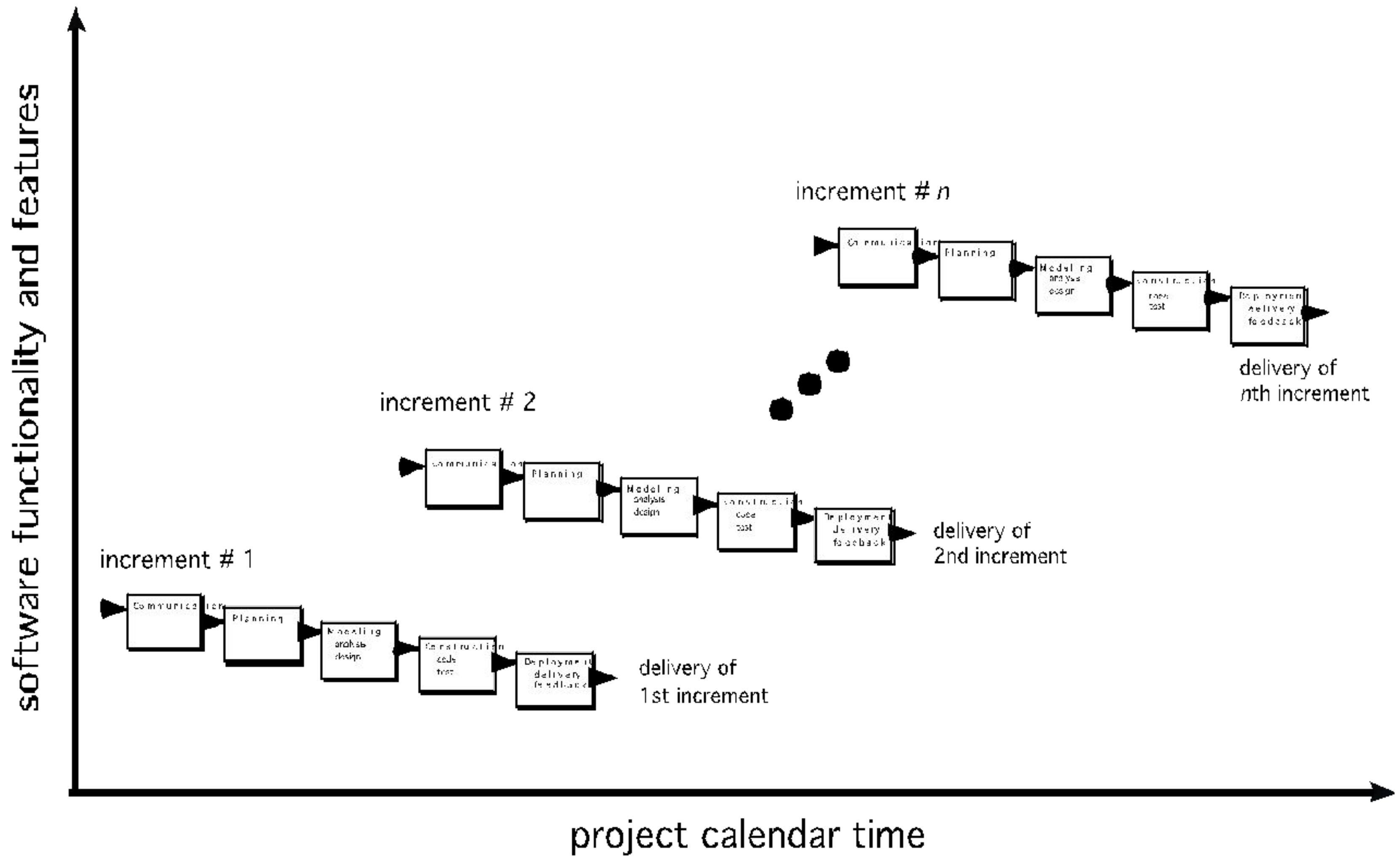
The Waterfall Model with feedback



V Model


Requirements modeling
Architectural Design
Component Design
Code Generation
Executable software
Unit Testing
Integration Testing
System Testing
Acceptance Testing

Incremental Process Model



Incremental Process Model

- Used when requirements are well understood
- Multiple independent deliveries are identified
- Work flow is in a linear (i.e., sequential) fashion <u>within</u> an increment and is staggered <u>between</u> increments
- Iterative in nature; focuses on an operational product with each increment
- Provides a needed set of functionality sooner while delivering optional components later
- Useful also when staffing is too short for a full-scale development

Evolutionary Process Model

Software system evolves over time as requirements often change as development proceeds.

Thus, a straight line to a complete end product is not possible. However, a limited version must be delivered to meet competitive pressure.

Usually a set of core product or system requirements is well understood, but the details and extension have yet to be defined.

You need a process model that has been explicitly designed to accommodate a product that evolved over time.

Evolutionary Process Model

It is iterative that enables you to develop increasingly more complete version of the software.

Two types are introduced, namely

Prototyping and Spiral models.

- Follows an evolutionary and iterative approach
- Used when requirements are <u>not</u> well understood
- Serves as a mechanism for identifying software requirements
- Focuses on those aspects of the software that are visible to the customer/user
- Feedback is used to refine the prototype

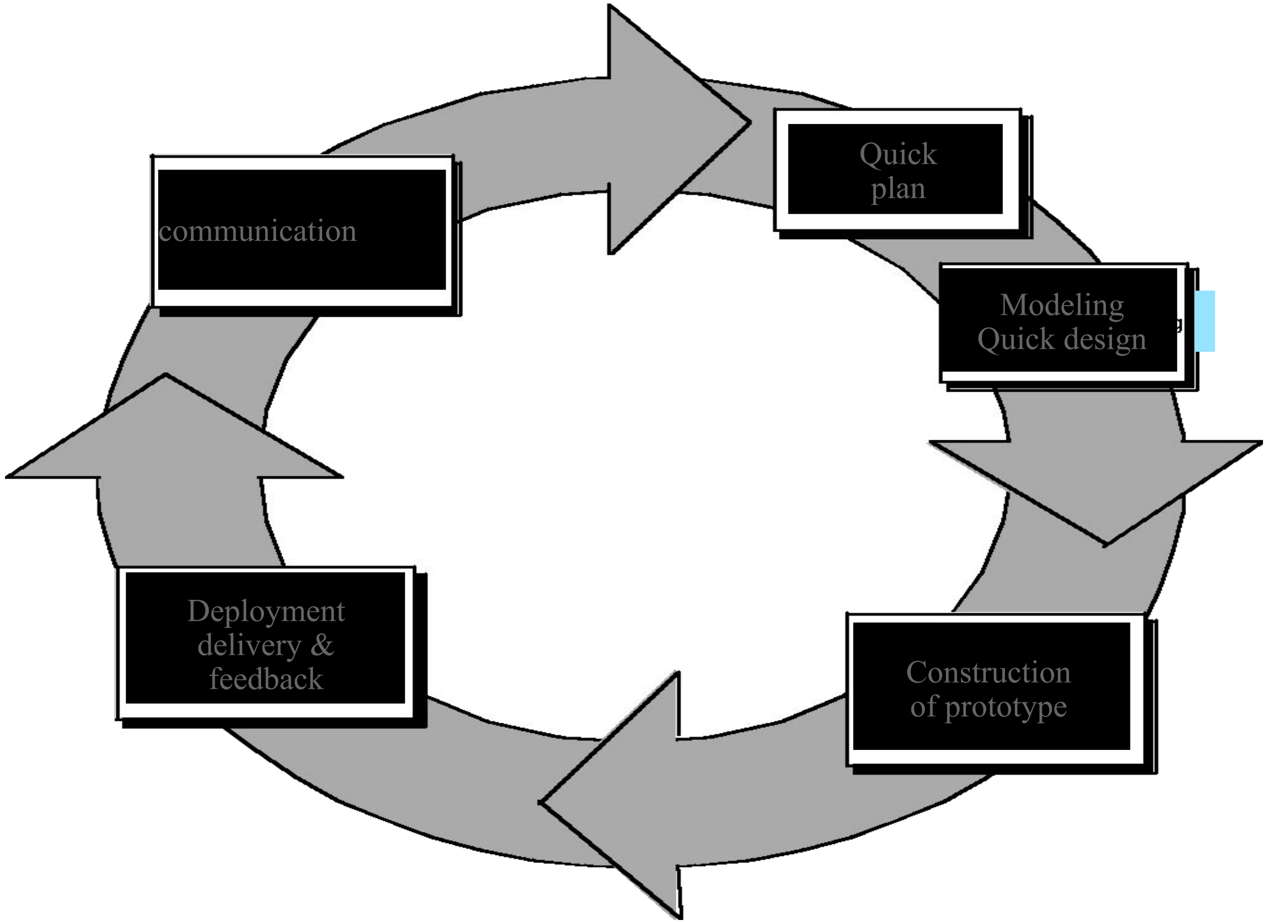
Prototyping

When to use: Customer defines objectives but does not specify requirements for functions and features. Or Developer may be unsure of the efficiency of an algorithm, the form that human computer interaction should take.

What step: Begins with communication by meeting with stakeholders to define the objective, identify whatever requirements are known. A quick plan for prototyping and modeling (quick design) occur. Stakeholder's comments will be used to refine requirements.

Both stakeholders and software engineers like the prototyping paradigm. Users get a feel for the actual system, and developers get to build something immediately.

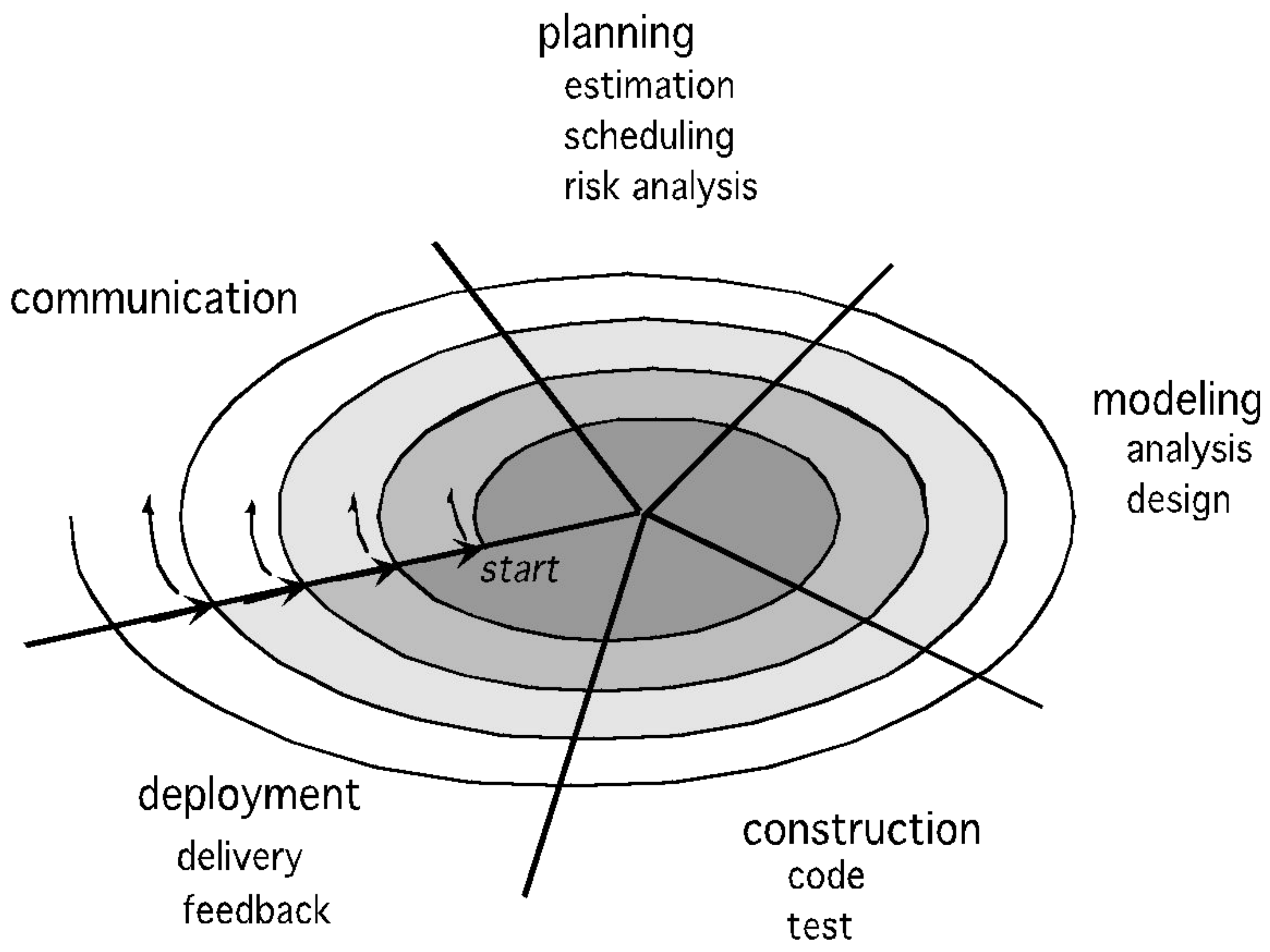
Prototyping



Spiral Model

It couples the iterative nature of **prototyping** with the controlled and systematic aspects of the **waterfall model** and is a risk-driven process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems.

Spiral Model



Spiral Model - features

One is cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk.

The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.

Three Concerns on Evolutionary Process

First concern is that prototyping poses a problem to project planning.

Because of the uncertain number of cycles required to construct the product.

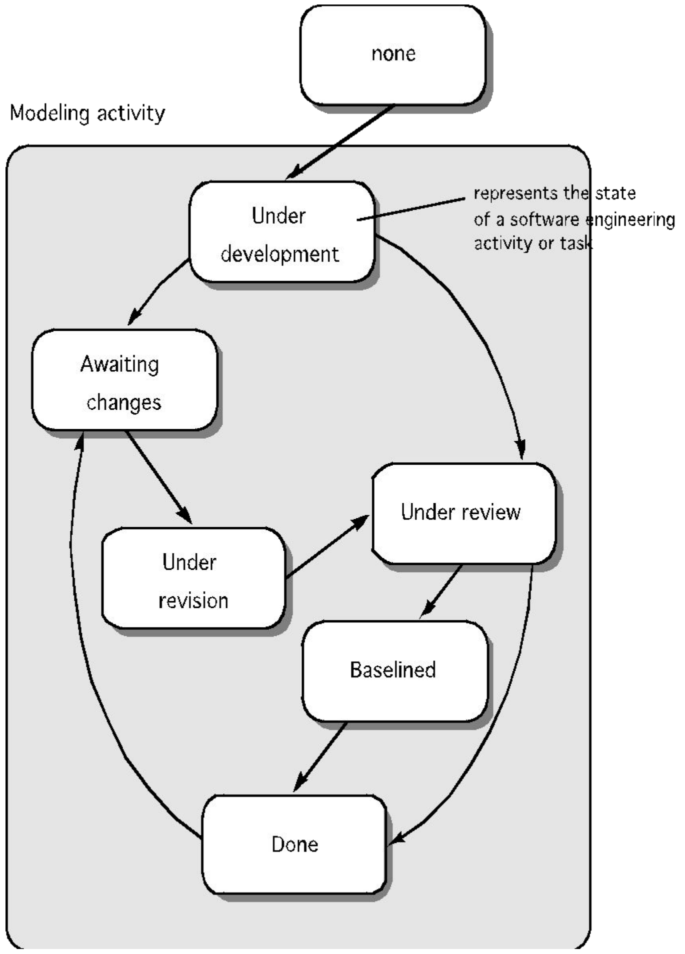
Second, it does not establish the maximum speed of the evolution.

If the evolution occur too fast, without a period of relaxation, it is certain that the process will fall into chaos. On the other hand if the speed is too slow then productivity could be affected.

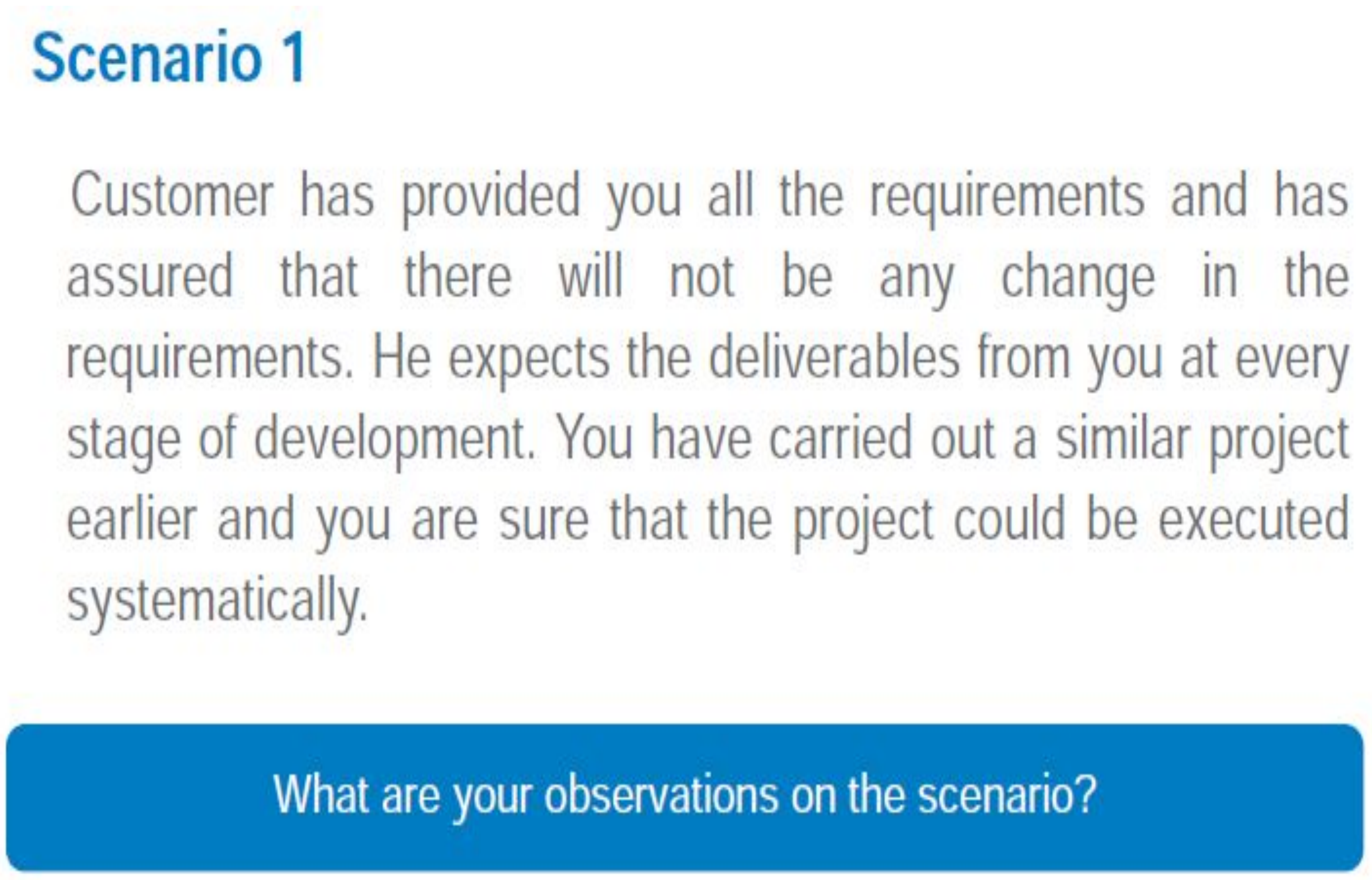
Third, software processes should be focused on flexibility and extensibility rather than on high quality.

We should prioritize the speed of the development over zero defects. Extending the development in order to reach high quality could result in a late delivery of the product when the opportunity niche has disappeared.

Concurrent Process Model



Exercise



Scenario 1

Customer has provided you all the requirements and has assured that there will not be any change in the requirements. He expects the deliverables from you at every stage of development. You have carried out a similar project earlier and you are sure that the project could be executed systematically.

What are your observations on the scenario?

Scenario 2

Customer wants you to start developing the software for a remote controlled electronic toy. Customer is not sure of all requirements of the product. She has provided you an initial set of requirements with which she wants to have a feel of the product. She has informed you that few more requirements will be provided later.

What are your observations on the scenario?

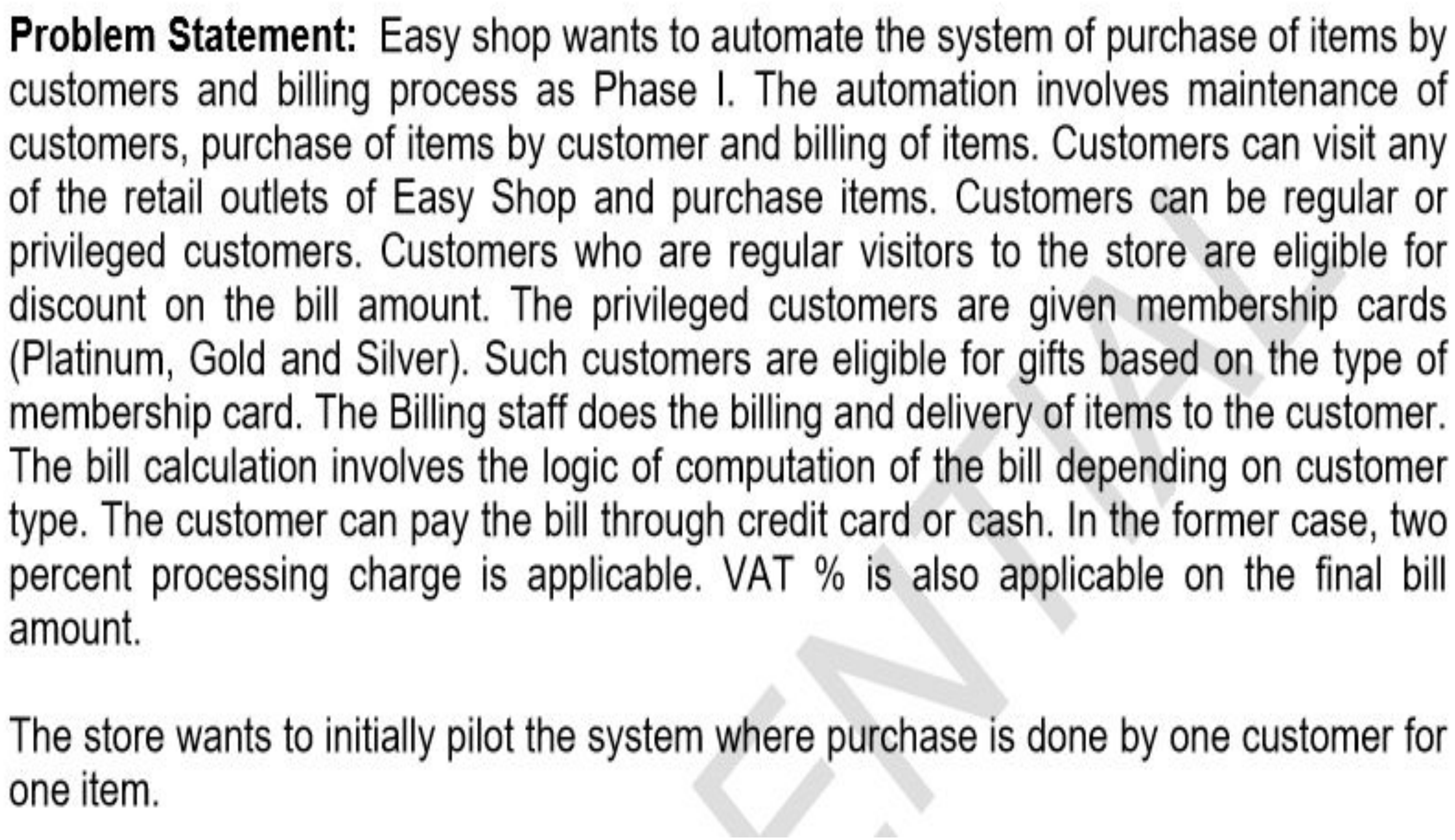
Scenario 3

Customer and the development team foresee many risks at every stage of software development. At each stage of development there are alternatives and you have to make right decisions. You and customer are in agreement that the project is not fixed budget project.

What are your observations on the scenario?

#	Scenario	Observations	Model
1	Customer has provided you all the requirements and has assured that there will not be any change in the requirements. He expects the deliverables from you at every stage of development. You have carried out a similar project earlier and you are sure that the project could be executed systematically.	• No change in requirements • Deliverables expected at every stage • Systematic execution	**Waterfall**
2	Customer wants you to start developing the software for a remote controlled electronic toy. Customer is not sure of all requirements of the product. She has provided you an initial set of requirements with which she wants to have a feel of the product. She has informed you that few more requirements will be provided later	• Complete requirements unavailable • Start development with a set of requirements • Initial feel of the product expected	**Incremental**
3	Customer and the development team foresee many risks at every stage of software development. At each stage of development there are alternatives and you have to make right decisions. You and customer are in agreement that the project is not fixed budget project	• Many risks are expected • There are alternatives at each stage • Not fixed budget project	**Spiral**

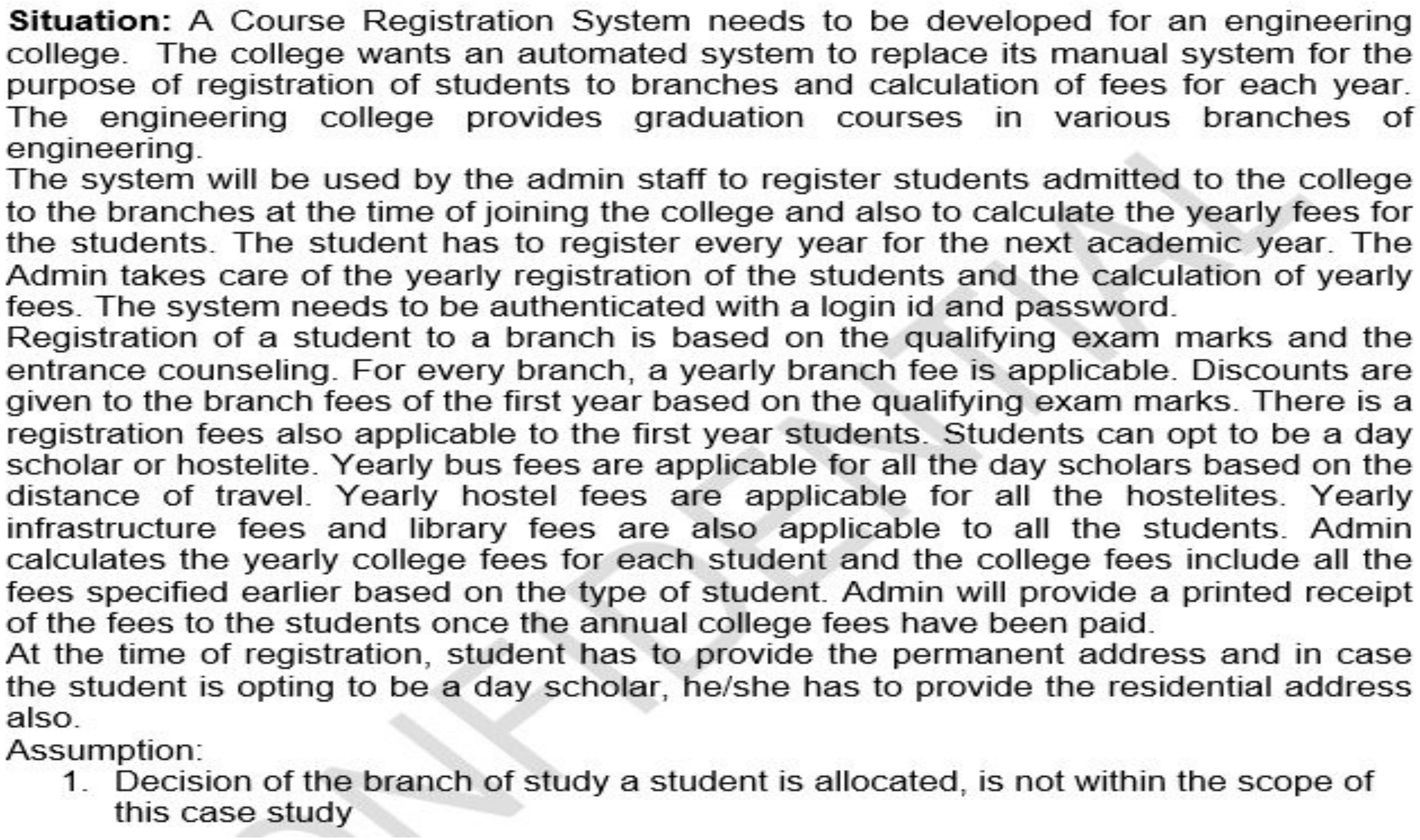
Case Study – Phase – I – Retail Application



Problem Statement: Easy shop wants to automate the system of purchase of items by customers and billing process as Phase I. The automation involves maintenance of customers, purchase of items by customer and billing of items. Customers can visit any of the retail outlets of Easy Shop and purchase items. Customers can be regular or privileged customers. Customers who are regular visitors to the store are eligible for discount on the bill amount. The privileged customers are given membership cards (Platinum, Gold and Silver). Such customers are eligible for gifts based on the type of membership card. The Billing staff does the billing and delivery of items to the customer. The bill calculation involves the logic of computation of the bill depending on customer type. The customer can pay the bill through credit card or cash. In the former case, two percent processing charge is applicable. VAT % is also applicable on the final bill amount.

The store wants to initially pilot the system where purchase is done by one customer for one item.

Case Study – Course Registration System



Situation: A Course Registration System needs to be developed for an engineering college. The college wants an automated system to replace its manual system for the purpose of registration of students to branches and calculation of fees for each year. The engineering college provides graduation courses in various branches of engineering.
The system will be used by the admin staff to register students admitted to the college to the branches at the time of joining the college and also to calculate the yearly fees for the students. The student has to register every year for the next academic year. The Admin takes care of the yearly registration of the students and the calculation of yearly fees. The system needs to be authenticated with a login id and password.
Registration of a student to a branch is based on the qualifying exam marks and the entrance counseling. For every branch, a yearly branch fee is applicable. Discounts are given to the branch fees of the first year based on the qualifying exam marks. There is a registration fees also applicable to the first year students. Students can opt to be a day scholar or hostelite. Yearly bus fees are applicable for all the day scholars based on the distance of travel. Yearly hostel fees are applicable for all the hostelites. Yearly infrastructure fees and library fees are also applicable to all the students. Admin calculates the yearly college fees for each student and the college fees include all the fees specified earlier based on the type of student. Admin will provide a printed receipt of the fees to the students once the annual college fees have been paid.
At the time of registration, student has to provide the permanent address and in case the student is opting to be a day scholar, he/she has to provide the residential address also.
Assumption:

1. Decision of the branch of study a student is allocated, is not within the scope of this case study

Thank You