Introduction to Software Engineering

Marwadi University

Computer Engineering Diploma

Unit 1:-
Introduction to Software Engineering

Software Engineering (09CE2402)

SOFTWARE

Software is:

(1)instructions (computer programs) that when executed provide desired features, function, and performance

(2)data structures that enable the programs to adequately manipulate information and

(3)documentation that describes the operation and use of the programs.

Software's Dual Role (Nature of Software)

- **Software is a product**
  - Transforms information - produces, manages, acquires, modifies, displays, or transmits information
  - Delivers computing potential of hardware and networks

    (WITHOUT S/W DIFFICULT TO OPERATE H/W)
- **Software is a vehicle(Process) for delivering a product**
  - Controls other programs (operating system)
  - Effects communications (networking software)
  - Helps build other software (software tools & environments)

Software Products

- **Generic Products:**
  - Stand-alone systems which are produced by a development organization and sold on the open market to any customer.(e.g. Microsoft Products)
- **Customized Products:**
  - Systems which are ordered by a specific customer and developed specially by some contractor.(e.g. College Mgmt. S/W)

Software Applications

(1) System Software:

- System Software is a set of programs that control and manage the operations of computer hardware. It also helps application programs to execute correctly.
- System Software are designed to control the operation and extend the processing functionalities of a computer system.
- System software makes the operation of a computer more fast, effective, and secure.

Ex. Compilers, operating system, drivers etc.

Software Applications

(2) Application Software :

- Application software consists of standalone programs that solve a specific business need.
- Application software is used to control the business function in real-time.
- Application Software acts as a mediator between the end-user and System Software.
- It is also known as an application package.
- This type of software is written using a high-level language like C, Java, VB. Net, etc.
- It is a user-specific and is designed to meet the requirements of the user.
- E.g. :Medical Store Mgmt. System

Difference B/W System S/W & Applications S/W

Sr. No	System Software	Application Software
1	System Software maintain the system resources and give the path for application software to run.	Application software is built for specific tasks.
2	Low level languages are used to write the system software.	While high level languages are used to write the application software.
3	It's General Purpose Software.	While it's a specific purpose software
4	Without system software, system can't run.	While without application software, system always runs.
5	System software runs when system is turned on and stop when system is turned off.	While application software runs as per the user's request.
6	Example of System software are operating system, etc.	Example of application software are Photoshop, VLC Player, etc.
7	System Software programming is complex than application software.	Application software programming is simpler as comparison to system software.

Software Applications

(3) Engineering /Scientific Software:

- Applications like space shuttle orbital dynamics, molecular biology and automated manufacturing.

Ex. Computer Aided Design (CAD), MATLAB, AUTOCAD, etc

Software Applications

(4) Embedded Software:

- It resides in read-only memory and is used to control products and systems
- Embedded software can perform limited functions.

Ex. keypad control for a microwave oven or timer

Software Applications

(5) Product line software:

- Designed to provide a specific capability for use by many different customers, product line software can focus on a limited and esoteric marketplace.

Ex. Word processing, spreadsheet, multimedia, etc.

Software Applications

(6) Web Applications:

- Web apps can be little more than a set of linked hypertext files.
- It evolving into sophisticated computing environments that not only provide standalone features, functions but also integrated with corporate database and business applications.

Software Applications

(7) Artificial Intelligence Software:

- AI software makes use of numerical & non-numerical algorithms to solve complex problems those are not amenable(easily controlled) to computation or straightforward analysis.

Ex. Robotics, expert system, game playing, etc.

Manufacturing vs. Development

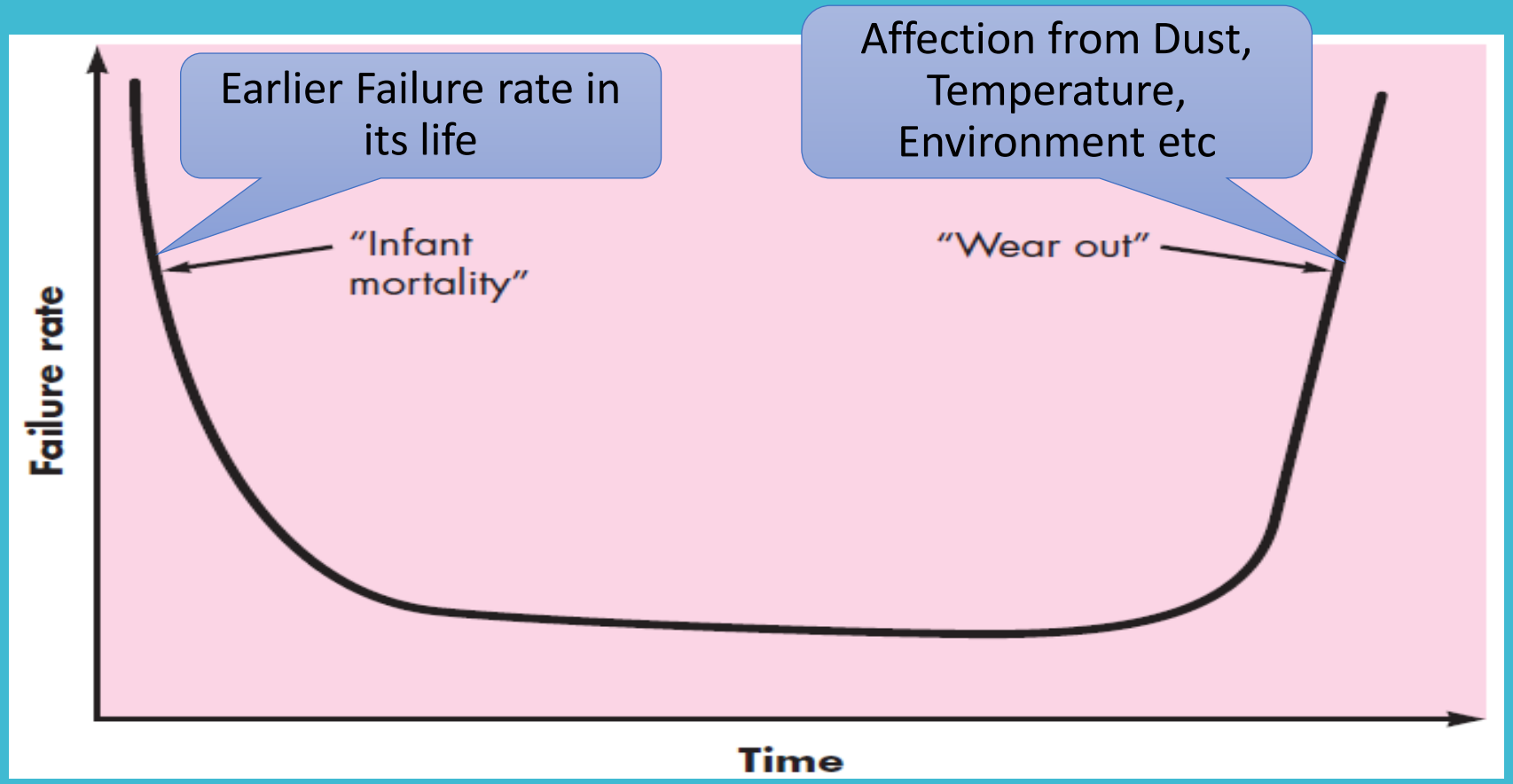
- Once a hardware product has been manufactured, it is difficult or impossible to modify. In contrast, software products are routinely modified and upgraded.
- In hardware, hiring more people allows you to accomplish more work, but the same does not necessarily hold true in software engineering.
- Unlike hardware, software costs are concentrated in design rather than production. (Once demo is ready then can produce multiple copy)

CHARACTERISTICS OF SOFTWARE

1] Software is developed or engineered

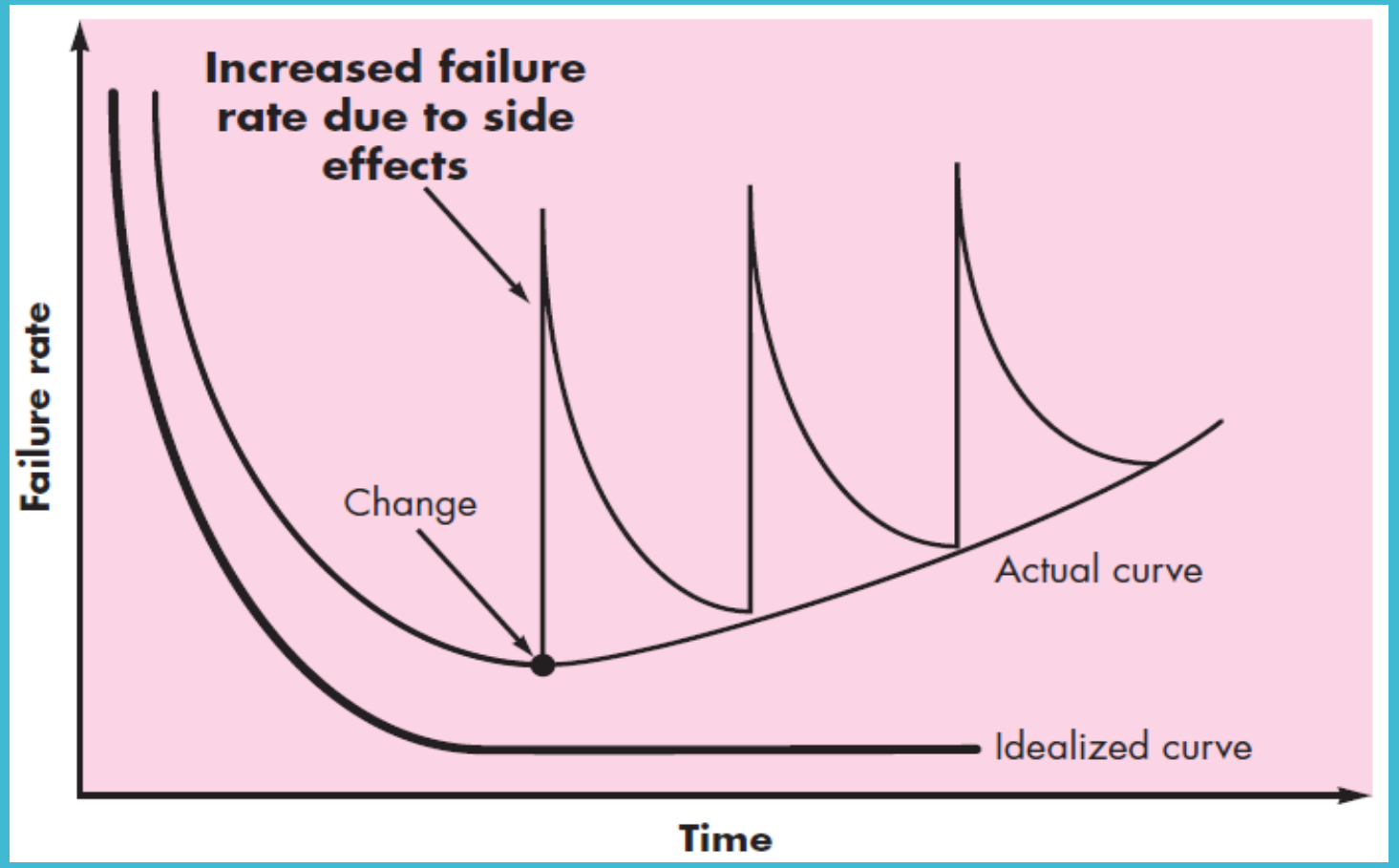
2] Software doesn't "wear out." (to use something a lot so that it no longer works)

Wear vs. Deterioration



Failure curve for Hardware

Wear vs. Deterioration



Software deteriorate over time

Hardware Vs. Software

Hardware	Software
▪ Manufactured ▪ Wears out ▪ Built using components ▪ Relatively simple	▪ Developed / Engineered ▪ Deteriorates(become progressively worse) ▪ Custom built ▪ Complex

Software Engineering

- Software Engineering is the science and art of building significant software systems that are:
  - 1) on time
  - 2) on budget
  - 3) with acceptable performance
  - 4) with correct operation.

Software Engineering

- A discipline whose aim is the production of quality software, delivered on time, within budget, and satisfying users' needs.
- Software engineering is concerned with theories, methods and tools for professional software development.

Software Engineering

- Software engineers should **adopt**
  1) **Systematic and organized** approach to their work
  2) Use **appropriate tools and techniques** depending on the problem to be solved
  3) The **development constraints and the resources available**
- **Challenge for Software Engineers** is to produce high quality software with **finite amount of resources & within a predicted schedule**

Software Myths

Software Myths

“Beliefs about software and the process used to build it”

- Software Myths – Management
- Software Myths – Customer
- Software Myths – Practitioner (Developer)

Management Myths

Management Myths

- The members of an organization can acquire all-the information, they require from a manual, which contains standards, procedures, and principles
- Standards may be outdated or incomplete,
- Developer may be unaware of it
- Developer rarely follow all the known standards

Management Myths

- If we get behind schedule, we can add more programmers and catch up- New workers take longer to learn about the project as compared to those already working on the project.
- If I decide to outsource the software project to a third party, I can just relax and let that firm build it.- Outsourcing software to a third party does not help the organization, which is incompetent in managing and controlling the software project internally. The organization invariably suffers when it out sources the software project.

Customer Myths

Customer Myths

- Brief requirement stated in the initial process is enough to start development; detailed requirements can be added at the later stages.
- Starting development with incomplete and ambiguous requirements often lead to software failure.
- Adding requirements at a later stage often requires repeating the entire development process.

Customer Myths

- Project requirements continually change, but changes can be easily accommodated because software is flexible.
- Incorporating change requests earlier in the development process costs lesser than those that occurs at later stages. This is because incorporating changes later may require redesigning and extra resources.

Practitioner (Developer) Myths

Practitioner's Myths

- Once we write the program and get it to work, our job is done.
- 50% to 70% of all the efforts are expended after the software is delivered to the user.
- The success of a software project depends on the quality of the product produced.
- The quality of programs is not the only factor that makes the project successful instead the documentation and software configuration also play a crucial role.

Practitioner's Myths

- The only product that is delivered after the completion of a project is the working program(s)

- The deliverables of a successful project includes not only the working program but also the documentation to guide the users for using the software.

- Software engineering requires unnecessary documentation, which slows down the project.

- Software engineering is about creating quality at every level of the software project. Proper documentation enhances quality which results in reducing the amount of rework.

Attributes of Software

Attributes of Good Software

- **(1) Maintainability (જાળવણી)**

  Software must evolve to meet changing needs;

- **(2) Dependability (નિર્ભરતા)**

  Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure.

  Malicious users should not be able to access or damage the system.

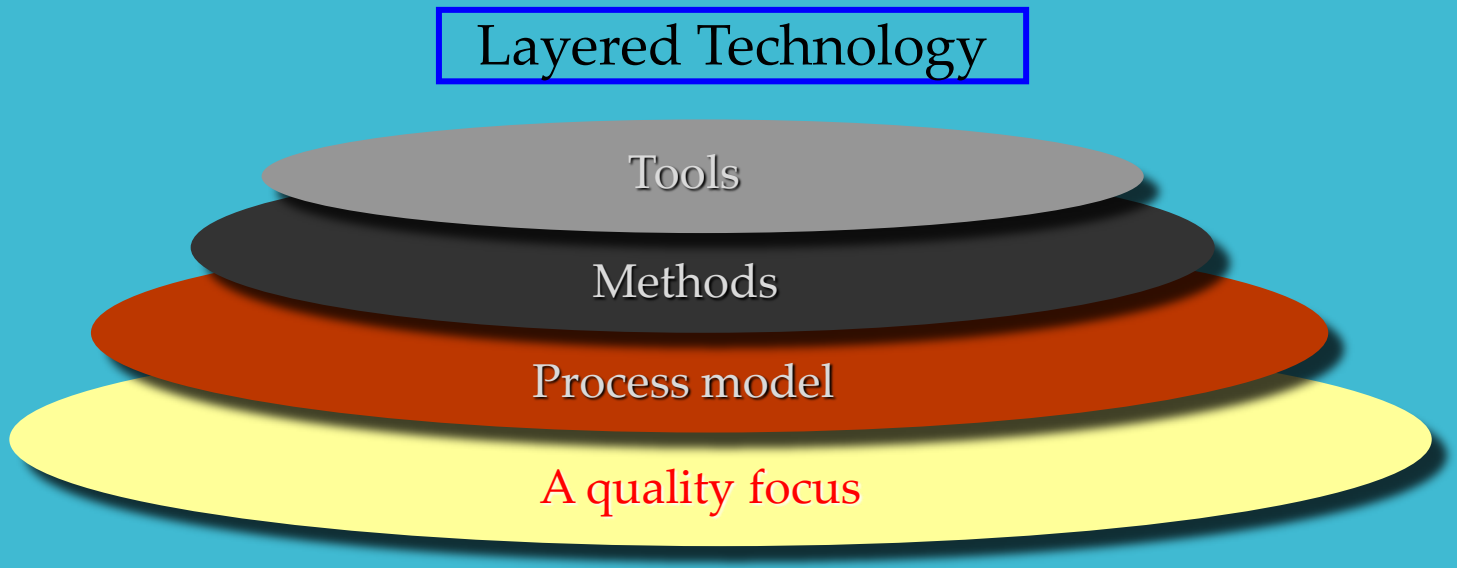
- **(3) Efficiency (કાર્યક્ષમતા)**

  Software should not make wasteful use of system resources.

- **(4) Acceptability (સ્વીકાર્યતા)**

  Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

Software Engineering – Layered Technology



Layered Technology

(1) A Quality Focus

The characteristics of good quality software are:

- Correctness of the functions required to be performed by the software.
- Maintainability of the software
- Integrity i.e. providing security so that the unauthorized user cannot access information or data.
- Usability i.e. the efforts required to use or operate the software.

Layered Technology

(2) Process

- It is the base layer or foundation layer for the software engineering.
- The software process is the key to keep all levels together.
- It defines a framework that includes different activities and tasks.
- In short, it covers all activities, actions and tasks required to be carried out for software development.
- What to do?
- How to do?

Layered Technology

(3) Methods

- The method provides the answers of all 'how-to' that are asked during the process.
- It provides the technical way to implement the software.
- It includes collection of tasks starting from communication, requirement analysis, analysis and design modelling, program construction, testing and support.

Layered Technology

(4) Tools

- Provide **automated or semi-automated** support for the process, methods and quality control.
- When tools are **integrated** so that information created by one tool can be used by another, a system for the support of software development, called ***computer-aided software engineering (CASE)***

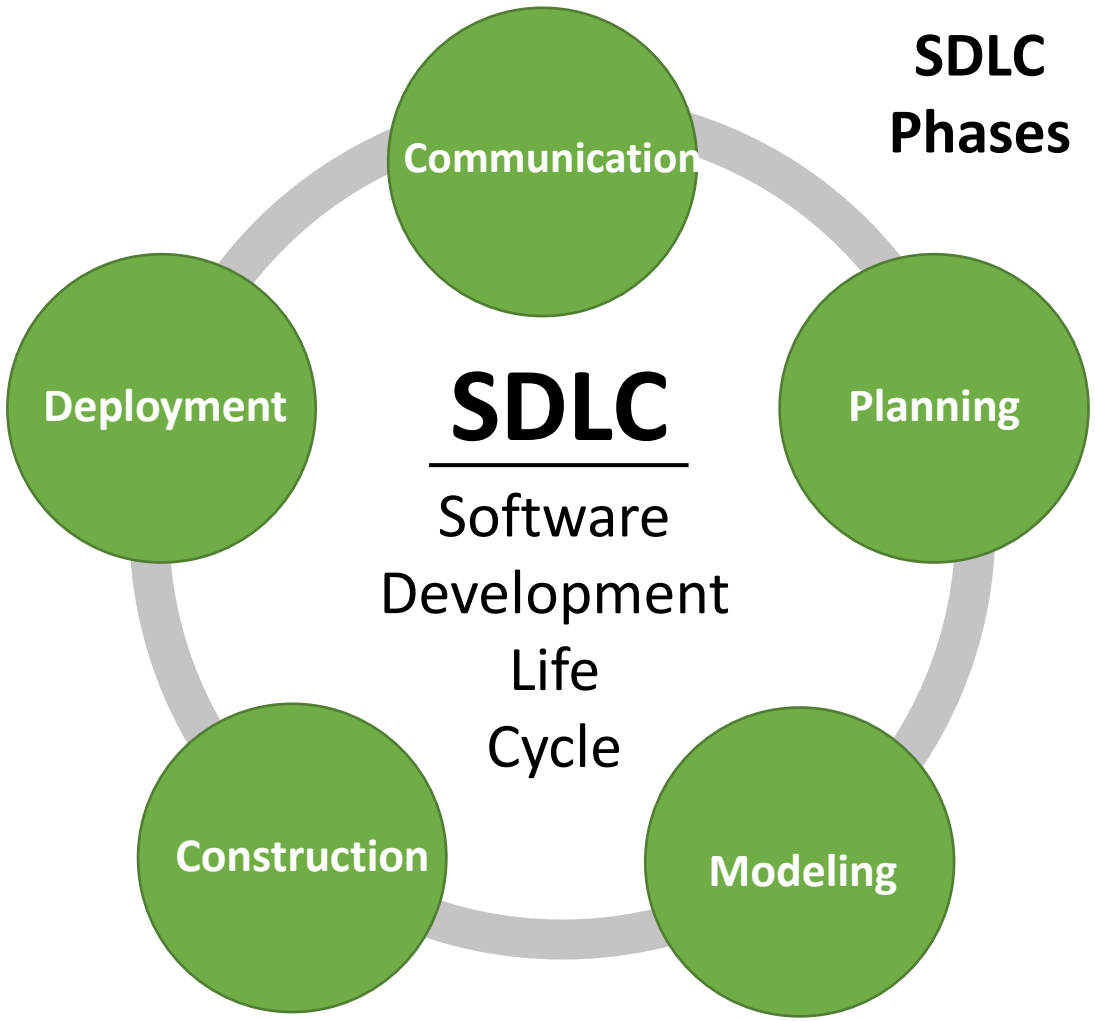
Software Process

- ✓Communication
- ✓Planning
- ✓Modeling
- ✓Construction
- ✓Deployment

Software Process Models

The **process model** is the abstract representation of process.

- Also known as **Software development life cycle (SDLC)** or Application development life cycle Models
- Process models **prescribe** a distinct set of **activities, actions, tasks and milestones (deliverables)** required to engineer high quality software.
- Process **models are not perfect,** but **provide roadmap** for software engineering work.
- Software models provide stability, control and organization to a process that if not managed can easily get out of control.
- Software process models are adapted (adjusted) to meet the needs of software engineers and managers for a specific project.



Software Process Models

Software Process Models

- ✓Linear Sequential Model **(Waterfall Model)**
- ✓Rapid Application Development **(RAD)** Model
- ✓Evolutionary Process Model
  - ✓**Prototyping**
  - ✓**Spiral Model**
- ✓ **Component Based Model**

Introduction to Software Engineering

Marwadi University

Computer Engineering Diploma

Unit 1:-
Introduction to Software Engineering

Software Engineering (09CE1402)

DEFINITIONS…

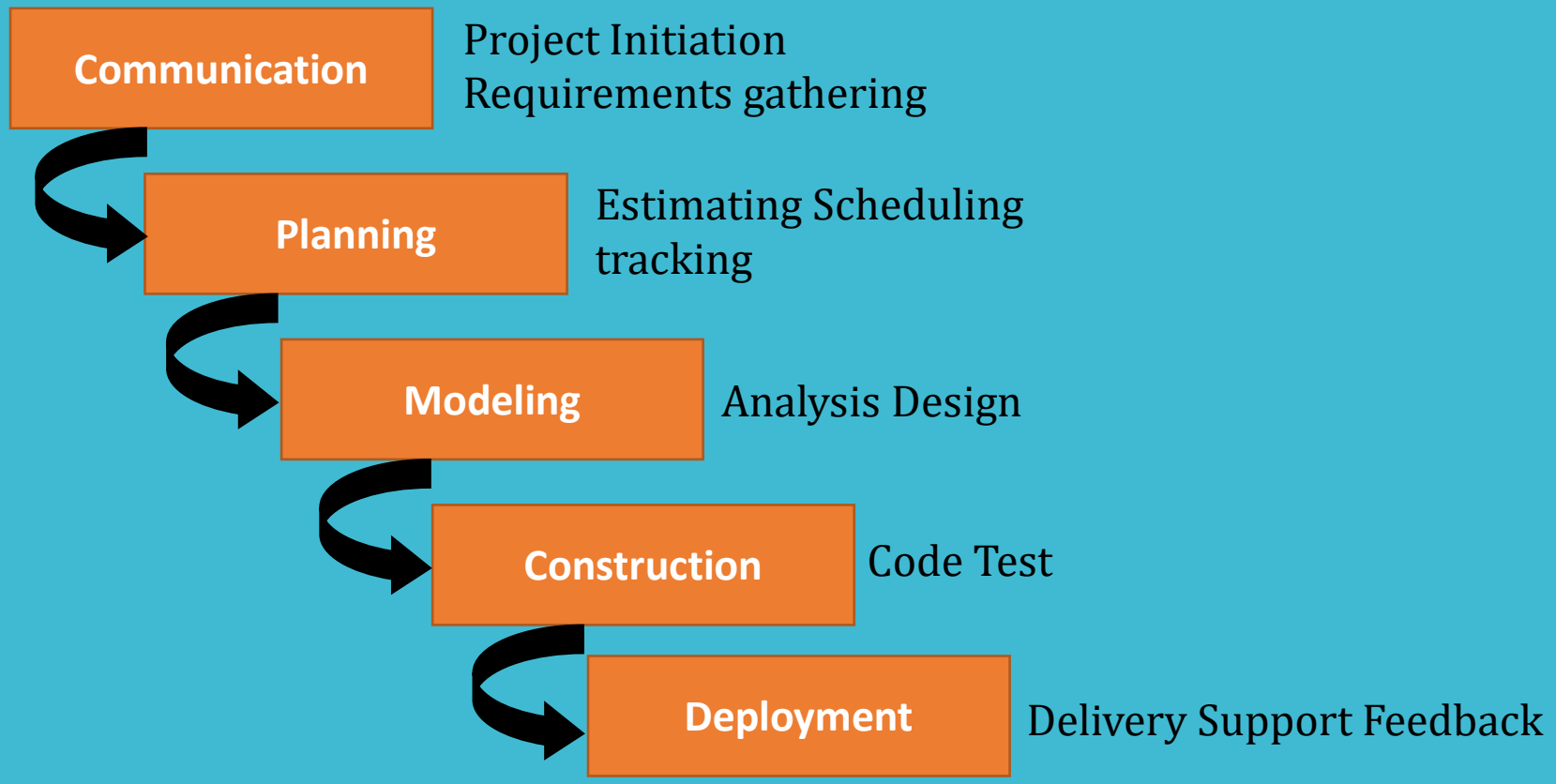
Definition of Software Engineering

- Software is a program or set of programs containing instructions which provide desired functionality.
- And Engineering is the processes of designing and building something that serves a particular purpose and find a cost effective solution to problems.
- ***Software Engineering*** *is a systematic approach to the design, development, operation, and maintenance of a software system.*

Linear Sequential Model

- The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model.
- In a waterfall model, each phase must be completed before the next phase can begin.
- Waterfall model is the earliest **SDLC (System Development Life Cycle)** approach that was used for software development.

Waterfall Model or Linear Sequential Model



Waterfall Model or Linear Sequential Model

- **Requirement Analysis and Definition:**

  ✓ Systems services, constraints and goals are defined by customers with system users.

- **Scheduling tracking:**

  ✓ Assessing progress against the project plan. Require action to maintain schedule.

- **System and Software Design:**

  ✓ It establishes overall system architecture. Software design involves fundamental system abstractions and their relationships.

Waterfall Model or Linear Sequential Model

- **Integration and system testing:**
  - ✓ The individual program unit or programs are integrated and tested as a complete system to ensure that the software requirements have been met.
  - ✓ After testing, the software system is delivered to the customer.
- **Operation and Maintenance:**
  - ✓ Normally this is the longest phase of the software life cycle.
  - ✓ The system is installed and put into practical use.
  - ✓ Maintenance involves correcting errors which were not discovered in earlier stages of the life-cycle.

PROS OF WATERFALL MODEL

1) Simple and easy to understand and use
2) Easy to manage due to the rigidity of the model .
3) Each phase has specific deliverables and a review process.
4) Phases are processed and completed one at a time.
5) Works well for smaller projects where requirements are very well understood.
6) Clearly defined stages.
7) Easy to arrange tasks.
8) Process and results are well documented.

LIMITATIONS OF WATERFALL MODEL

- The nature of the **requirements will not change** very much during development
- The model implies that you should attempt to **complete a given stage before moving** on to the next stage
  - Does not account for the fact that requirements constantly change.
  - It also means that customers can not use anything until the entire system is complete.
- The model implies that once the product is finished, **everything else is maintenance.**
- Surprises at the end are very expensive
- Some teams sit ideal for other teams to finish
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.

Problem

1. Real projects are **rarely follow the sequential model.**
2. Difficult for the customer **to state all the requirement in advance.**
3. **Assumes patience from customer** - working version of program will not available until programs not getting change fully.

When to use waterfall model

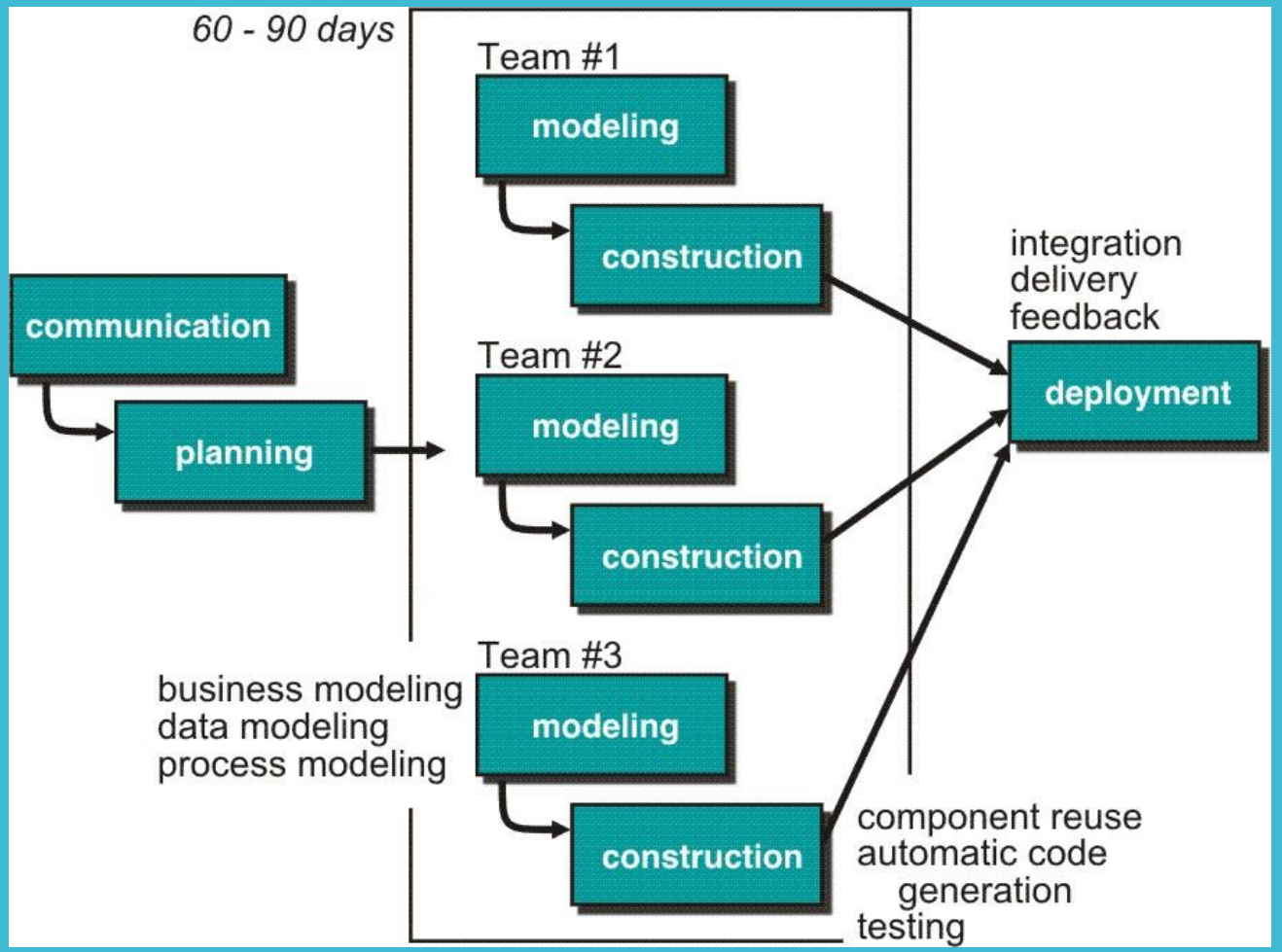
1) Requirements are very well known, clear and fixed
2) Product definition is stable
3) Technology is understood
4) There are no ambiguous requirements
5) Sufficient resources with required expertise are available freely
6) The project is short

RAD Model

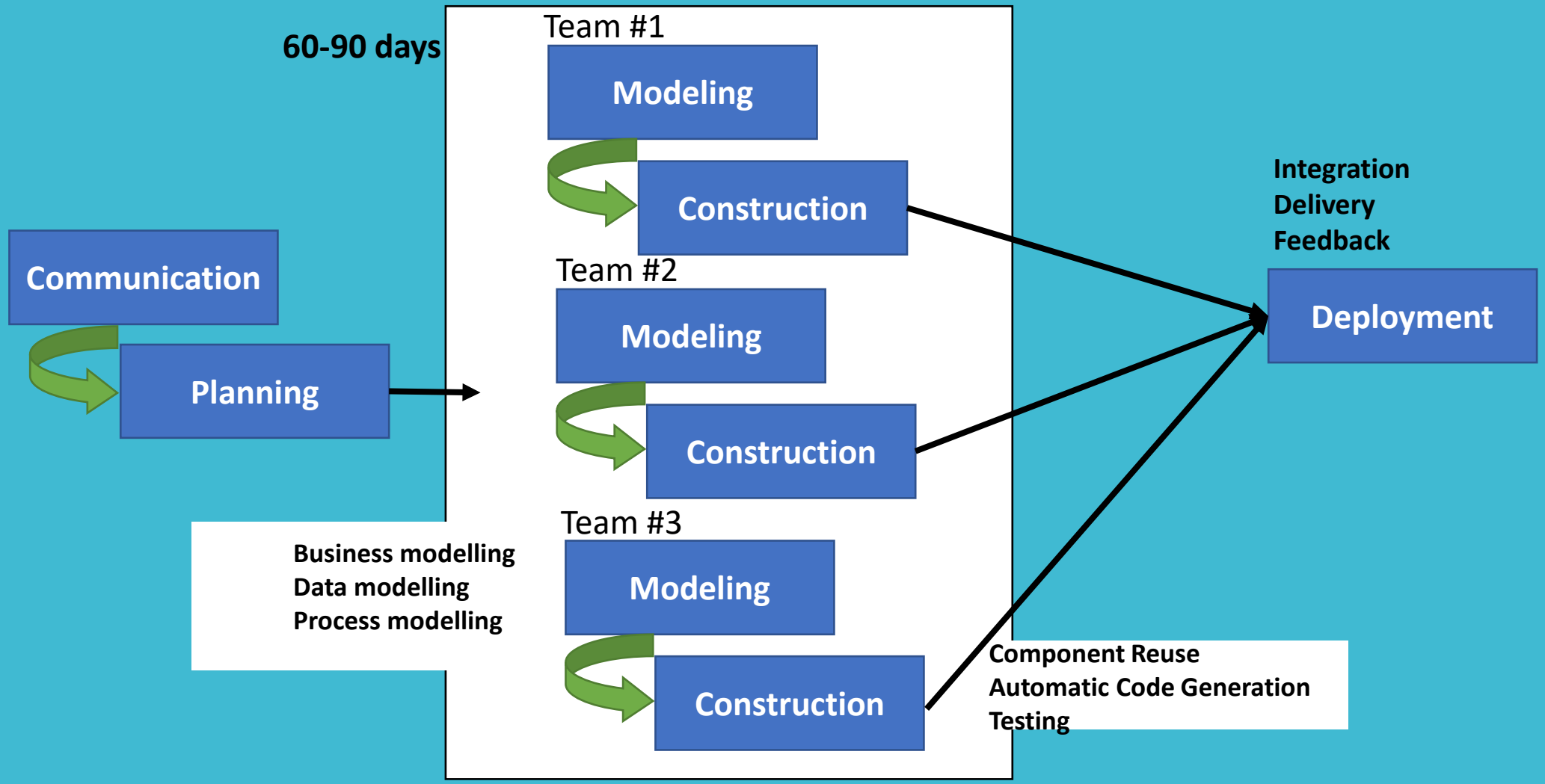
RAD Model

- RAD stands for **Rapid Application Development.**
- This model is similar to waterfall model which achieves the high speed development using component based construction.
- To develop the fully functional system within the short period of time using this model, it is necessary to understand the requirements fully and to have a restricted project scope.

Rapid Application Development (RAD) Model



Rapid Application Development (RAD) Model



RAD MODEL

(1) Communication – To understand business problem.

(2) Planning – To prepare documentations of all the requirements, schedule, etc.

(3) Modeling –

(a) Business Modeling – Information flow among business is working.

Ex. What kind of information drives?

Who is going to generate information?

From where information comes?

(b) Data Modeling – Information refine into set of data objects that are needed to support business.

(c) Process Modeling – Data object transforms to information flow necessary to implement business.

RAD MODEL

- **Deployment –** Deliver to customer basis for subsequent iteration.

❑ “**High speed**” edition of linear sequential model.

❑ If requirement are well understood and project scope is constrained then it enable development team to create “ fully functional system” within a very short time period.

Cons of RAD model

1) It requires multiple teams or large number of people to work on the project.
2) This model requires heavily committed developer and customers. It may fails, if commitment is lacking.
3) It requires heavy resources.
4) Lack of proper modularization may lead to project failure. Performance can be effected.

Evolutionary Process Models

Evolutionary Process Model

- Produce an increasingly more complete version of the software with each iteration.
- Evolutionary Models are **iterative.**
- Evolutionary models are:
  - **Prototyping**
  - **Spiral Model**

Prototype Model

Evolutionary Process Model- Prototyping Model

- Before carrying out the development of the actual software, **a prototype of the system is built.**
- It's a toy implementation of the software.
- It exhibits limited functional capabilities and inefficient performance compared to actual software.
- Prototype is used with several shortcuts. These shortcuts might involve using inefficient, inaccurate or dummy functions.

Prototyping Model

- The purpose of prototype is to illustrate the input data formats, messages, reports and interactive dialogues to the customer.
- **The prototype model is very useful in developing the GUI part of the software.**
- A developed prototype can help developer to examine the technical issues associated with the product development.
- Some issues are response time of a hardware controller, efficiency of an algorithms used.

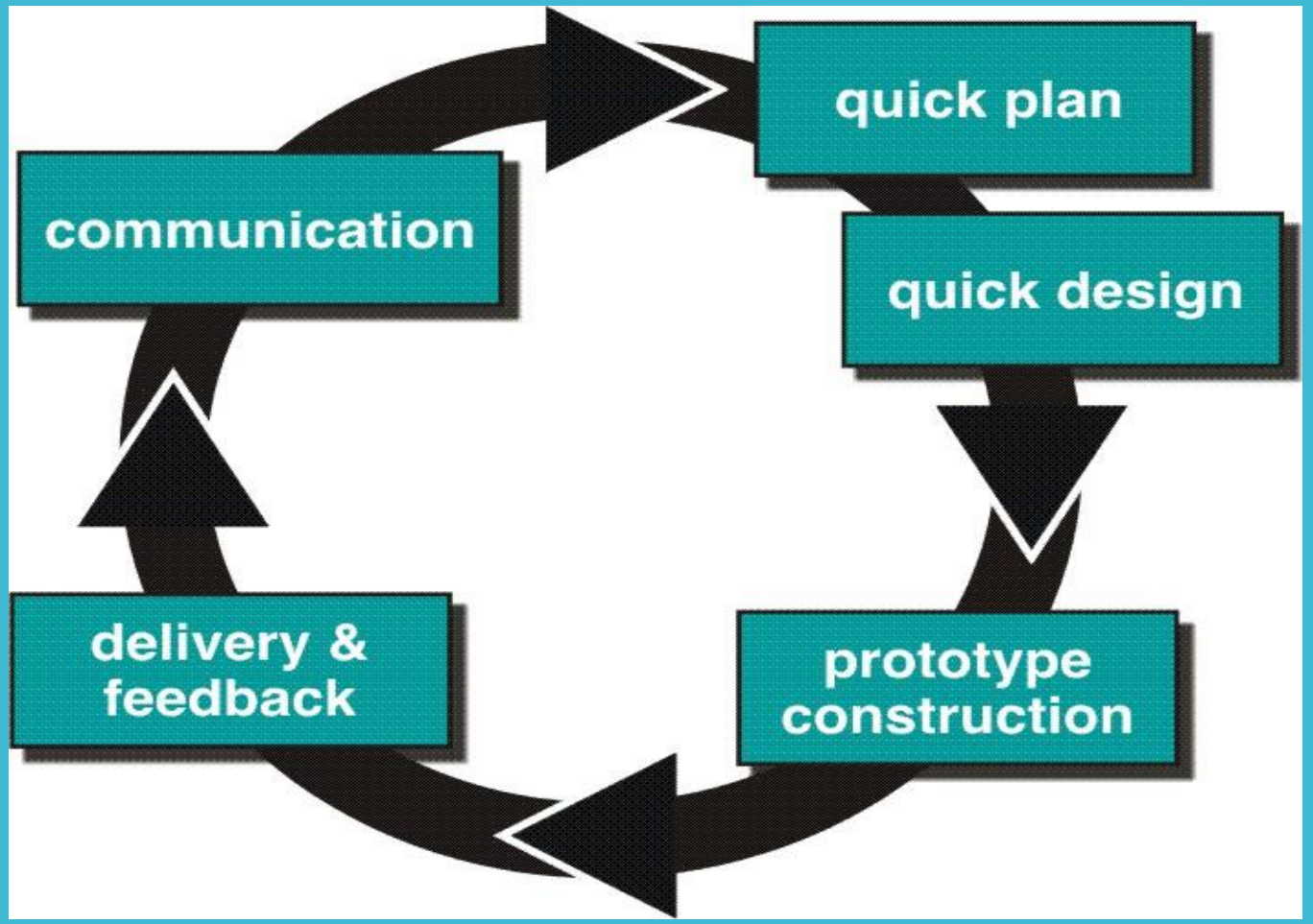
Prototyping Model

Phases of prototyping model:

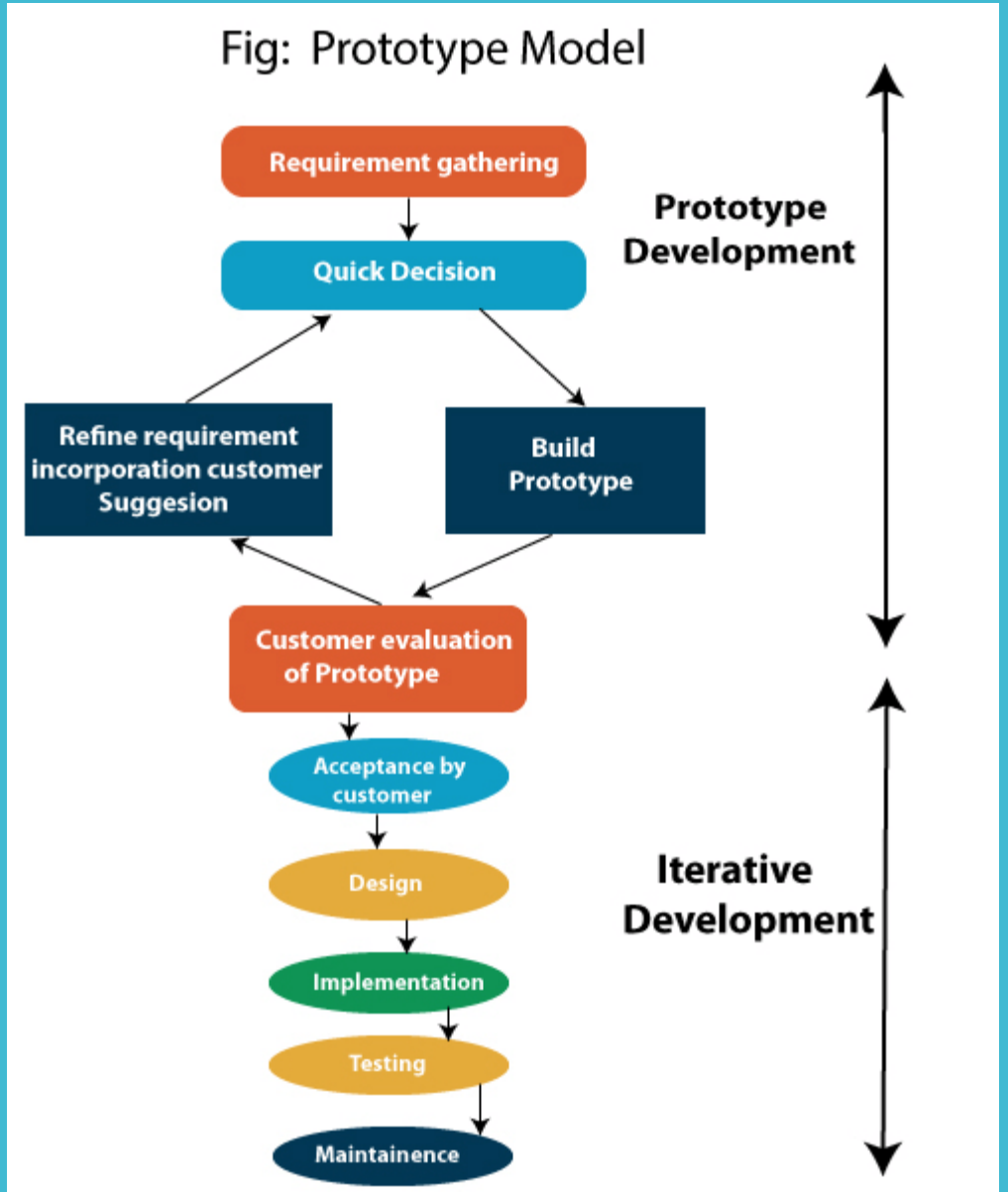
- ✓ Initial requirements gathering
- ✓ Quick design
- ✓ Build prototype
- ✓ Customer evolution of prototype
- ✓ Refine requirements incorporating customer suggestions

- **This phases are continued till the acceptance of customer.**
- After that we follow the Design, Implement, Test and Maintenance phases.

Prototyping Model



Prototyping Model



Prototyping Model

Problem Areas

- Customer demand that **“a few fixes”** be applied to make the prototype a working product, due to that **software quality** suffers as a result.
- Developer often makes implementation in order to get a prototype working quickly without considering other factors in mind like OS, Programming language, etc.

Advantages Prototype Model

1) Reduce the risk of incorrect user requirement
2) Good where requirement are changing/uncommitted
3) Regular visible process aids management
4) Support early product marketing
5) Reduce Maintenance cost.
6) Errors can be detected much earlier as the system is made side by side.

Disadvantages Prototype Model

1) An unstable/badly implemented prototype often becomes the final product.
2) Require extensive customer collaboration
   - Costs customer money
   - Needs committed customer
   - Difficult to finish if customer withdraw
   - May be too customer specific, no broad market
3) Difficult to know how long the project will last.
4) Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5) Prototyping tools are expensive.
6) Special tools & techniques are required to build a prototype.
7) It is a time-consuming process.

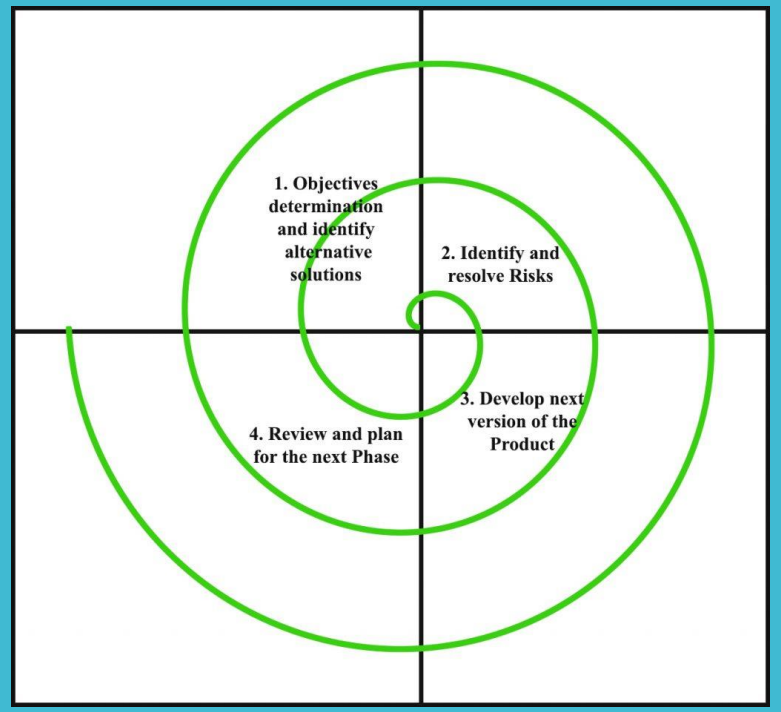
When to Use Prototype Model?

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.

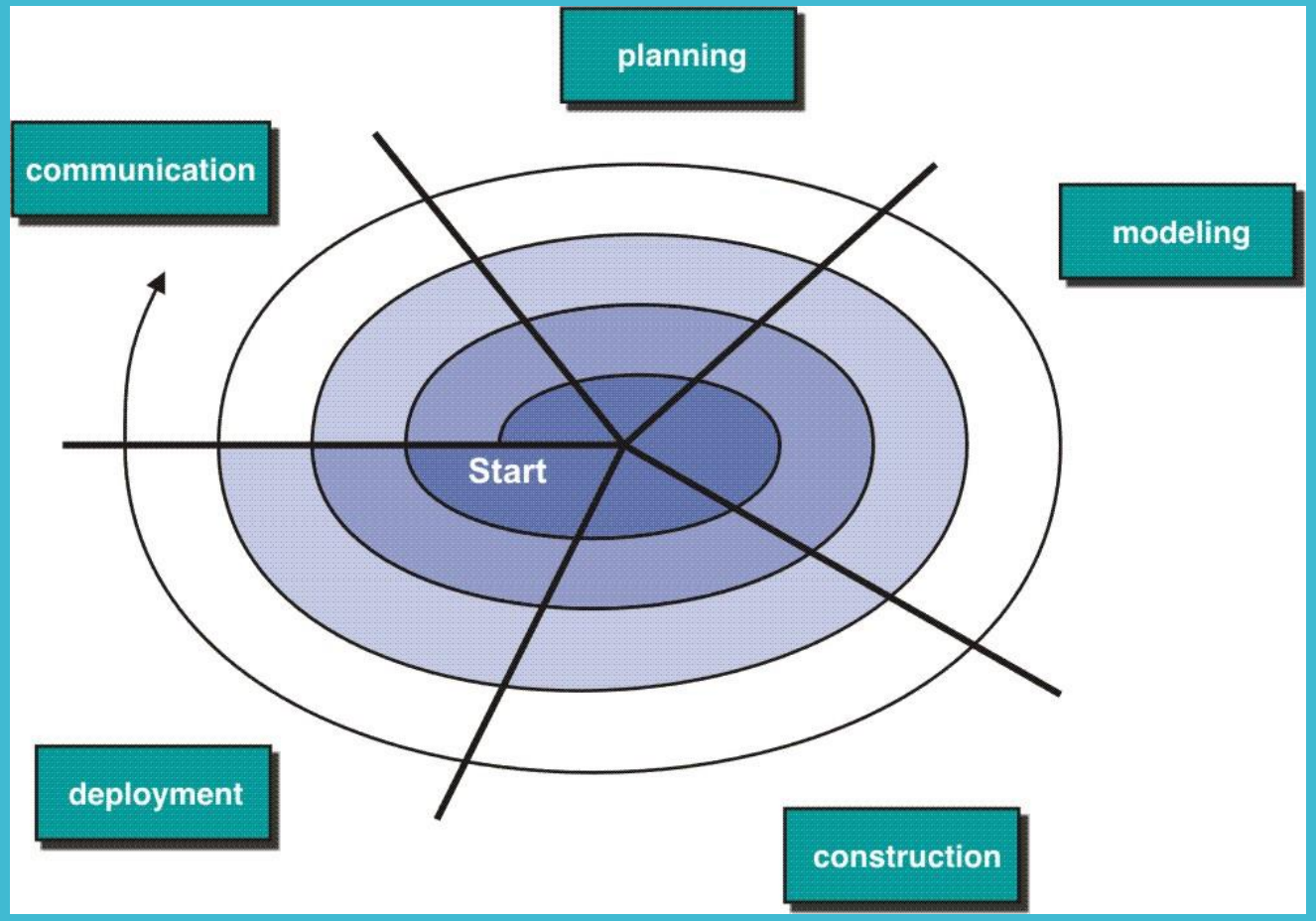
Spiral Model

Evolutionary Model: Spiral Model

- Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling.
- In its diagrammatic representation, it looks like a spiral with many loops.
- The exact number of loops of the spiral is unknown and can vary from project to project.
- **Each loop of the spiral is called a Phase of the software development process.**
- The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks.



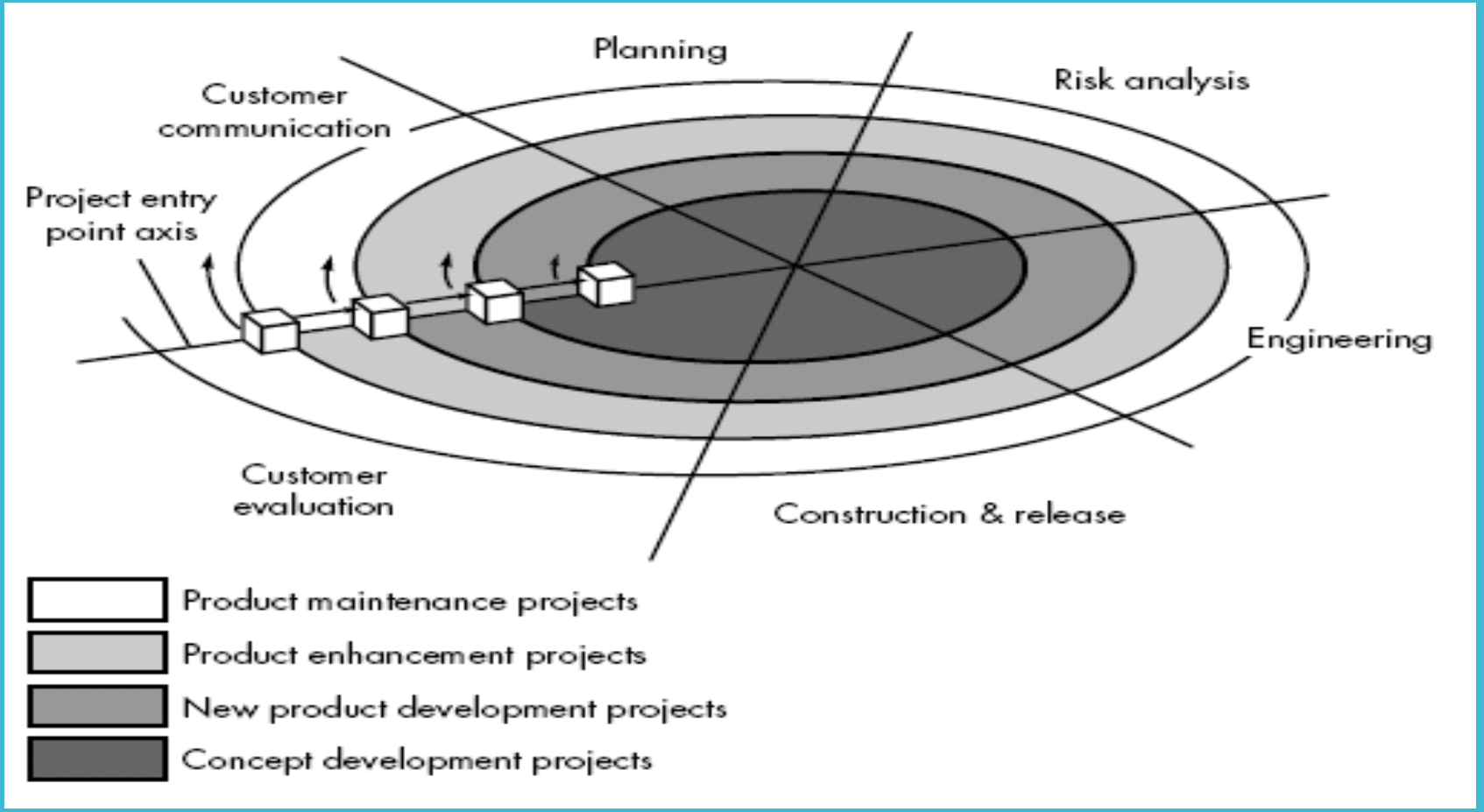
Evolutionary Model: Spiral Model



Spiral Model

- As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.
- The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Spiral Model



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

Advantages Spiral Model

- **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
- **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
- **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Disadvantages of Spiral Model

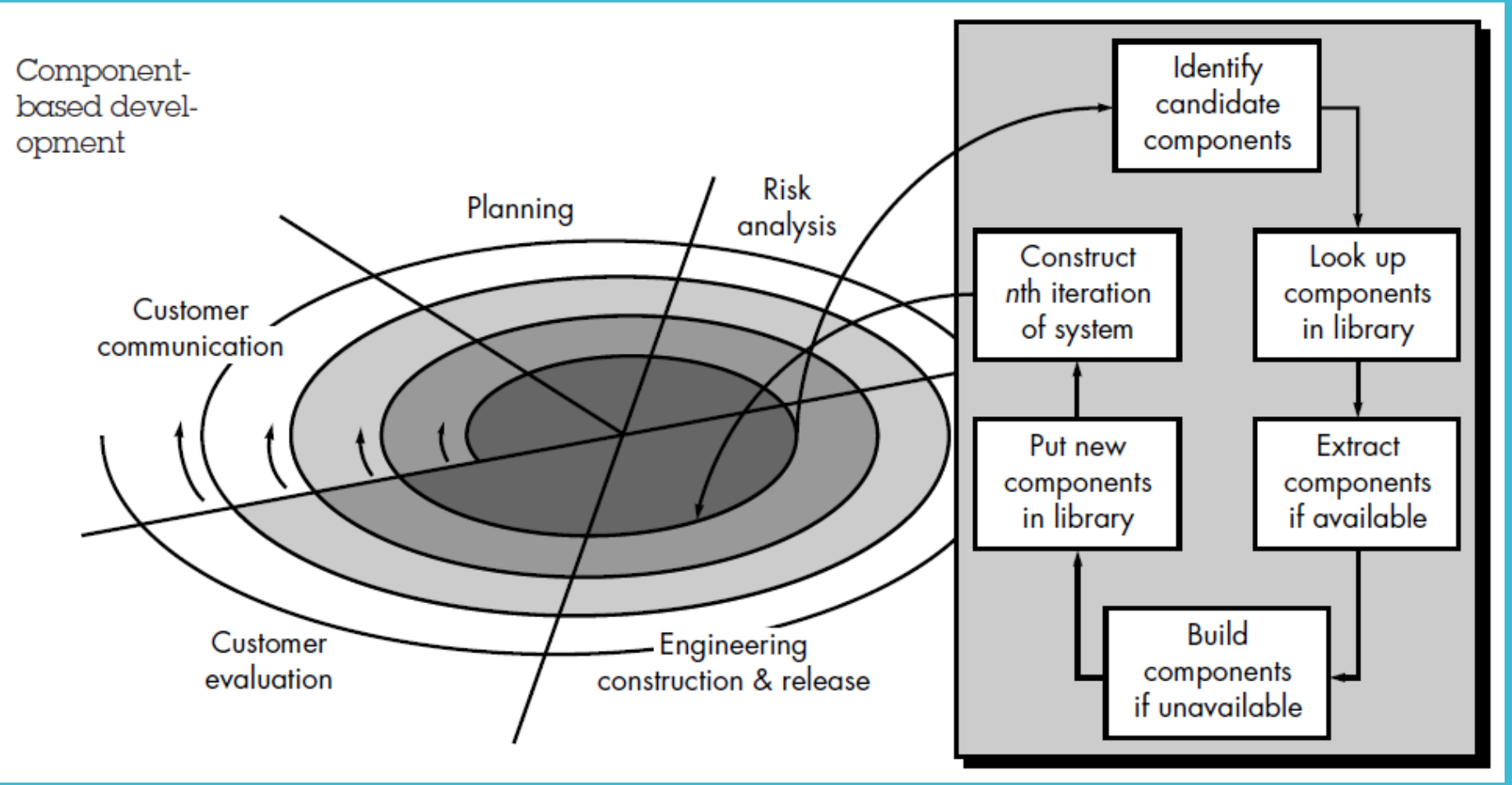
- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Too much dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.
- **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

Component Based Development Model

Component-based Development Model

- Component based development is a software system development methodology where the system is developed **using reusable software components.**
- Component based development aims at improved efficiency, performance and quality of the system by recycling components

Component-based Development Model



Component-based Development Model

- **Consists of the following process steps**
    1. Available component-based products are researched and evaluated for the application domain
    2. Component integration issues are considered
    3. A software architecture is designed to accommodate the components
    4. Comprehensive testing is conducted to ensure proper functionality
- Relies on a robust component library
- Capitalizes on software reuse, which leads to documented savings in project cost and time

THANK YOU!!!!