SUBJECT CODE : 3150711

As per New Syllabus of

GUJARAT TECHNOLOGICAL UNIVERSITY

Semester - V (CE / CSE) (Professional Elective - I)

SOFTWARE ENGINEERING

Anuradha A. Puntambekar

M.E. (Computer)

Formerly Assistant Professor in

P.E.S. Modern College of Engineering,

Pune

TECHNICAL PUBLICATIONS

SINCE 1993

An Up-Thrust for Knowledge

SOFTWARE ENGINEERING

Subject Code : 3150711

Semester - V (Computer Engineering / Computer Science & Engineering) (Professional Elective - I)

First Edition : August 2020

Published by :

TECHNICAL PUBLICATIONS
SINCE 1993
An Up-Thrust for Knowledge

Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :
Yogiraj Printers & Binders
Sr.No. 10/1A,
Ghule Industrial Estate, Nanded Village Road,
Tal. - Haveli, Dist. - Pune - 411041.

ISBN 978-81-946628-2-2

9788194662822

9788194662822 [1]

Course 18

PREFACE

The importance of **Software Engineering** is well known in various engineering fields. Overwhelming response to my books on various subjects inspired me to write this book. The book is structured to cover the key aspects of the subject **Software Engineering**.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All the chapters in the book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of the subject.

Representative questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

I wish to express my profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by my whole family. I wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Author

A. A. Puntambekar

Dedicated to God.

SYLLABUS

Software Engineering - 3150711

Credits	Examination Marks				Total Marks
C	Theory Marks		Practical Marks		
	ESE (E)	PA (M)	ESE (V)	PA (I)	
04	70	30	30	20	150

1. Introduction to Software and Software Engineering

The Evolving Role of Software, Software: A Crisis on the Horizon and Software Myths, Software Engineering: A Layered Technology, Software Process Models, The Linear Sequential Model, The Prototyping Model, The RAD Model, Evolutionary Process Models, Agile Process Model, Component-Based Development, Process, Product and Process. **(Chapter - 1)**

2. Agile Development

Agility and Agile Process model, Extreme Programming, Other process models of Agile Development and Tools. **(Chapter - 2)**

3. Managing Software Project

Software Metrics (Process, Product and Project Metrics), Software Project Estimations, Software Project Planning (MS Project Tool), Project Scheduling & Tracking, Risk Analysis &Management (Risk Identification, Risk Projection, Risk Refinement , Risk Mitigation). **(Chapter 3)**

4. Requirement Analysis and Specification

Understanding the Requirement, Requirement Modeling, Requirement Specification (SRS), Requirement Analysis and Requirement Elicitation, Requirement Engineering. **(Chapter - 4)**

5. Software Design

Design Concepts and Design Principal, Architectural Design, Component Level Design (Function Oriented Design, Object Oriented Design), User Interface Design, Web Application Design. **(Chapter - 5)**

6. Software Coding and Testing

Coding Standard and coding Guidelines, Code Review, Software Documentation, Testing Strategies, Testing Techniques and Test Case, Test Suites Design, Testing Conventional Applications, Testing Object Oriented Applications, Testing Web and Mobile Applications, Testing Tools (Win runner, Load runner). **(Chapter - 6)**

7. Quality Assurance and Management

Quality Concepts and Software Quality Assurance, Software Reviews (Formal Technical Reviews), Software Reliability, The Quality Standards: ISO 9000, CMM, Six Sigma for SE, SQA Plan. **(Chapter - 7)**

8. Software Maintenance and Configuration Management

Types of Software Maintenance, Re-Engineering, Reverse Engineering, Forward Engineering, The SCM Process, Identification of Objects in the Software Configuration, Version Control and Change Control **(Chapter - 8)**

9. DevOps

Overview, Problem Case Definition, Benefits of Fixing Application Development Challenges, DevOps Adoption Approach through Assessment, Solution Dimensions,

What is DevOps?, DevOps Importance and Benefits, DevOps Principles and Practices, 7 C's of DevOps Lifecycle for Business Agility, DevOps and Continuous Testing, How to Choose Right DevOps Tools, Challenges with DevOps Implementation, Must Do Things for DevOps, Mapping My App to DevOps - Assessment, Definition, Implementation, Measure and Feedback **(Chapter - 9)**

10. Advanced Topics in Software Engineering

Component-Based Software Engineering, Client/Server Software Engineering, Web Engineering, Reengineering, Computer-Aided Software Engineering, Software Process Improvement, Emerging Trends in software Engineering. **(Chapter - 10)**

TABLE OF CONTENTS

1 Introduction to Software and Software Engineering

Syllabus

The evolving role of software, Software : A crisis on the horizon and software myths, Software engineering : A layered technology, Software process models, The linear sequential model, The prototyping model, The RAD model, Evolutionary process models, Agile process model, Component-based, Development, Process, Product and process.

Contents

1.1 Introduction

Software is nothing but a collection of computer programs and related documents that are intended to provide desired features, functionalities and better performance.

Software products may be

1. **Generic** - That means developed to be sold to a range of different customers.
2. **Custom** - That means developed for a single customer according to their specification.

1.2 Evolving Role of Software

The evolving role of software means changing role of software. Basically any software appears in two roles :

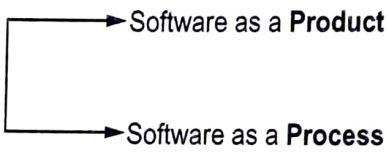

Fig. 1.2.1

Being a **product** the role of software can be recognised by its computing potentials, hardware capabilities and accessibility of network computers by the local hardware. Being a **product** it also acts as an **information transformer** i.e. producing, managing, modifying and conveying the source of information.

Being a **process** the software acts as a vehicle for driving the product. In this role the duty of software is to **control the computer** or to establish **communications** between the computers. Thus software plays dual role. It is both a product and a vehicle for delivering a product.

The role of software is significantly changing over past decade. There are many **factors affecting the role of software** and those are -

- Changes in computer architectures (Right from Pentium I to supercomputers)
- Improvement in hardware performance.
- Vast increase in amount of memory
- Wide variety of input and outputs(Ranging from simple text to multimedia videos)

Following table presents the different roles of software in different era of computing.

Era of computing	Software
Early Years	Batch processing Custom software
Second Era	Multi user Real time systems Database systems Product software

Third Era	Distributed systems
Forth Era	Object oriented systems Expert systems Parallel computing Network computers
Fifth Era	Web technologies mobile computing

- As 1990 began, **Toffler** described power shift. That is-old systems such as Government, educational, economic, military make use of computers performing the assigned tasks. And software lead to **democratization of knowledge.**

Democratization of knowledge means use of computers at all the public places and development of useful software applications allowed more and more people to access it. New technologies and improved user experience increasing number of users of the computers. Due to increasing scale, consumers have greater access to use and purchase technologically sophisticated products and services.

But this evolutionary role of software brings some crucial problems. Here are some sample **problems** encountered due to evolution on software

1. Advances in hardware demand for more capable software.
2. Ability to build new programs can not meet the demand for new programs and such programs are not sufficient for business and market needs.
3. Vast use of computer based systems brings less use of manpower and ultimately society becomes more dependant on machine and not on man.
4. Constant struggle for high reliability and quality software.

Review Questions

1. *Explain the role of software in democratization of knowledge.*
2. *Explain the evolving role of software.*

1.3 What is Software Engineering ?

"Software engineering is a discipline in which theories, methods and tools are applied to develop professional software."

In software engineering a systematic and organized approach is adopted. Based on the nature of problem and development constraints various tools and techniques are applied in order to develop quality software.

Role of Software Engineer

The software engineer has to adopt systematic and organized approach in order to produce high quality software. He is also responsible for selecting the most appropriate method for software development.

1.4 Software Characteristics

GTU : Winter-2012, 2013, Marks 7

Software development is a logical activity and therefore it is important to understand basic characteristics of software. Some important characteristics of software are :

- **Software is engineered, not manufactured**

Software development and hardware development are two different activities. A good design is a backbone for both the activities. Quality problems that occur in hardware manufacturing phase can not be removed easily. On the other hand, during software development process such problems can be rectified. In both the activities, developers are responsible for producing qualitative product.

- **Software does not ware out**

In early stage of hardware development process the failure rate is very high because of manufacturing defects. But after correcting such defects the failure rate gets reduced. The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies (extreme temperature, dusts, and vibrations).

On the other hand software does not get affected from such environmental maladies. Hence ideally it should have an *"idealized curve"*. But due to some **undiscovered errors** the failure rate is high and drops down as soon as the errors get corrected. Hence in failure rating of software the *"actual curve"* is as shown in Fig. 1.4.1.

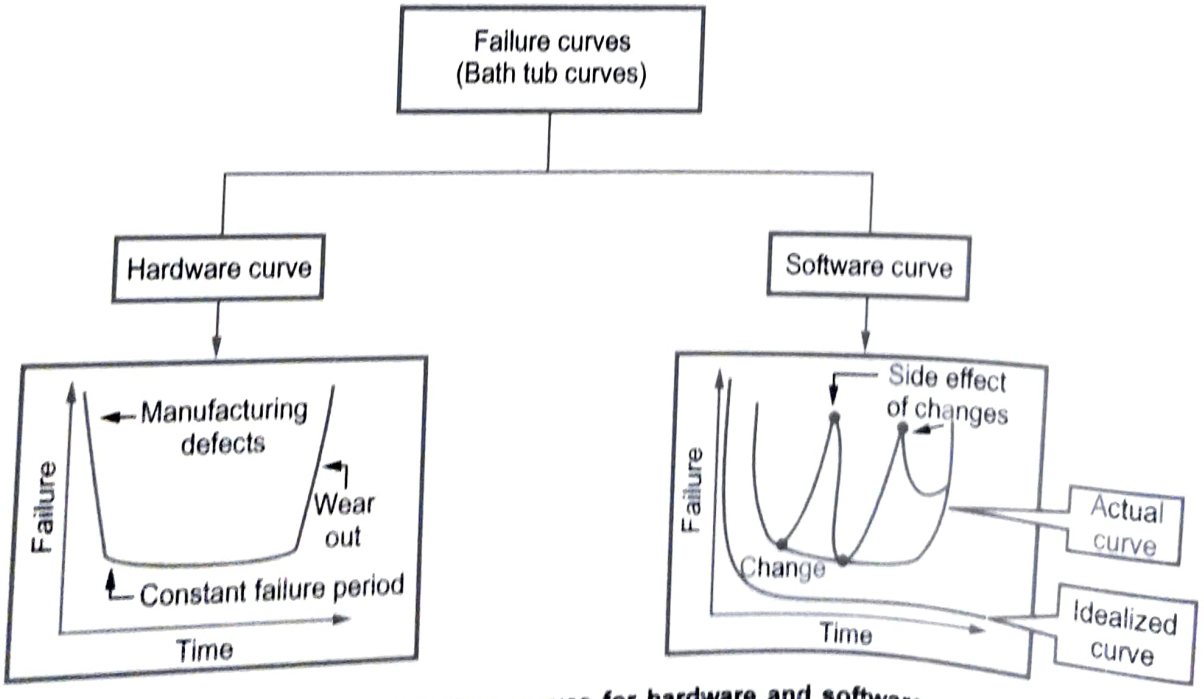

Fig. 1.4.1 Failure curves for hardware and software

During the life of software if any change is made, some **defects** may get **introduced**. This causes failure rate to be high. Before the curve can return to original steady state another change is requested and again the failure rate becomes high. Thus the failure curve looks like a spike. Thus frequent changes in software cause it to deteriorate.

Another issue with software is that there are ***no spare parts for software***. If hardware component wears out it can be replaced by another component but it is not possible in case of software. Therefore **software maintenance** is **more difficult** than the hardware maintenance.

- **Most software is custom built rather than being assembled from components**

While developing any hardware product firstly the circuit design with desired functioning properties is created. Then required hardware components such as ICs, capacitors and registers are assembled according to the design, but this is not done while developing software product. Most of the software is custom built.

However, now the software development approach is getting changed and we look for reusability of software components. It is practiced to reuse algorithms and data structures. Today software industry is trying to make library of reusable components. **For example :** in today's software, GUI is built using the reusable components such as message windows, pull down menus and many more such components. The approach is getting developed to use in-built components in the software. This stream of software is popularly known as *component engineering*.

Comparison between Hardware and Software Product characteristics

Sr. No.	Hardware product characteristics	Software product characteristics
1.	Hardware products are manufactured. The quality problems that occur in hardware manufacturing phase can not be removed easily.	Software is engineered and not manufactured. During software development process, various problems can be identified and rectified.
2.	In early stage of hardware development process, the failure rate is very high because of manufacturing defects. But after correcting such defects the failure rate gets reduced. The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies.	On the other hand, software does not get affected from the environmental maladies. But due to some undiscovered errors the failure rate is high and drops down as soon as the errors get corrected.
3.	There are spare parts available for the hardware components.	There are no spare parts for software components to replace.
4.	The hardware unit is assembled from components	Most software is custom built rather than being assembled from components.

Review Questions

1. *What is software engineering ? What is the role of software engineer? Compare hardware and software characteristics.* **GTU : Winter-2012, Marks 7**
2. *Explain the difference between software and hardware characteristics.* **GTU : Winter-2013, Marks 4**

1.5 Software : Crisis on the Horizon

The software crisis means the decisive time or turning point that software developers encounter during software development. Hence **software crisis** represent various problems that are faced by the software developers during software development process.

To understand software crisis consider following problem that is often faced by software industry.

Software cost is getting increased tremendously day-by-day. The software purchase expenses are higher than the hardware purchase. This is becoming worrying trend over the years. Following graph shows the ratio of h/w and s/w cost vs years -

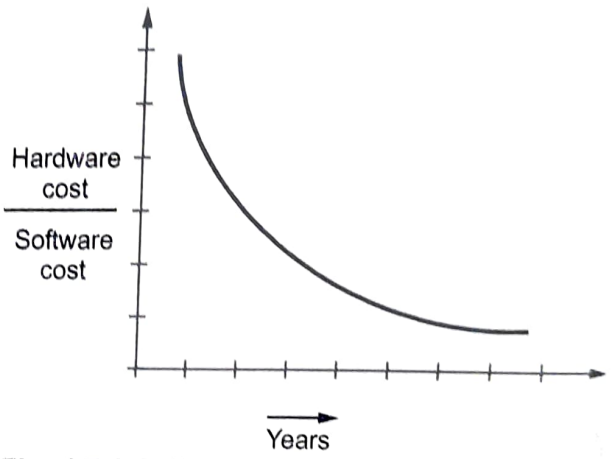

Fig. 1.5.1 Software cost : A software crisis

This graph shows that cost of software is increasing rapidly than the hardware cost. Following are the symptoms of present software crisis -

1. Day-by-day, software purchase cost is getting more than the hardware purchase cost. Hence major part of budget of any software industry is on software purchase.
2. Software products are difficult to alter, maintain, enhance, debug or modify.
3. Software resources are not being used optimally.
4. User requirements are often evolving and cannot be satisfied fully.
5. Software product not being reliable.
6. Many time software products get crashed on occurrence of specific condition.
7. Delivery of software product within specified budget and on scheduled time.

Various factors that have contributed to make software crisis

Following are some factors that bring the software crisis -

i) Increasing size or volume of software.

ii) Lowered productivity or quality improvement.

iii) Lack of skilled staff.

iv) Inadequate software training.

v) Growing demand for more software.

Solutions to present software crisis -

The most effective solutions to present software crisis can be i) use and spread of software engineering practices among software engineers and ii) further enhancement in software engineering desciplines.

1.6 Software Myths

There are some misbeliefs in the software industry about the software and process of building software. For any software developer it is a must to know such beliefs and reality about them. Here are some typical myths -

- **Myth :** Using a collection of standards and procedures one can build software. (Management Myth)
  Reality : Eventhough we have all standards and procedures with us for helping the developer to build software, it is not possible for software professionals to build desired product. This is because - the collection which we have should be complete, it should reflect modern techniques and more importantly it should be adaptable. It should also help the software professional to bring quality in the product.
- **Myth :** Add more people to meet deadline of the project. (Management Myth)
  Reality : Adding more people in order to catch the schedule will cause the reverse effect on the software project i.e. software project will get delayed. Because, we have to spend more time on educating people or informing them about the project.
- **Myth :** If a project is outsourced to a third party then all the worries of software building are over. (Management Myth)
  Reality : When a company needs to outsource the project then it simply indicates that the company does not know how to manage the projects. Sometimes, the outsourced projects require proper support for development.
- **Myth :** Even if the software requirements are changing continuously it is possible to accommodate these changes in the software. (Customer Myth)
  Reality : It is true that software is a flexible entity but if continuous changes in the requirements have to be incorporated then there are chances of introducing more and more errors in the software. Similarly, the additional resources and more design modifications may be demanded by the software.
- **Myth :** We can start writing the program by using general problem statements only. Later on using problem description we can add up the required functionalities in the program. (Customer Myth)
  Reality : It is not possible each time to have comprehensive problem statement.

We have to start with general problem statements; however by proper communication with customer the software professionals can gather useful information. The most important thing is that the problem statement should be unambiguous to begin with.

- **Myth :** Once the program is running then its over! (Practitioner's Myth)
  Reality : Even though we obtain that the program is running major part of work is after delivering it to customer.
- **Myth :** Working program is the only work product for the software project. (Practitioner's Myth)
  Reality : The working program/software is the major component of any software project but along with it there are many other elements that should be present in the software project such as documentation of software, guideline for software support.
- **Myth :** There is no need of documenting the software project; it unnecessarily slows down the development process. (Practitioner's Myth)
  Reality : Documenting the software project helps in establishing ease in use of software. It helps in creating better quality. Hence documentation is not wastage of time but it is a must for any software project.

1.7 Software Engineering : A Layered Technology

GTU : Winter-2011, 2014, Marks 7, Winter-2017, 2018, Marks 3, Summer-2012, 2019, Marks 4

- Software engineering is a layered technology. Any software can be developed using these layered approaches. Various layers on which the technology is based are **quality focus layer, process layer, methods layer, tools layer.**

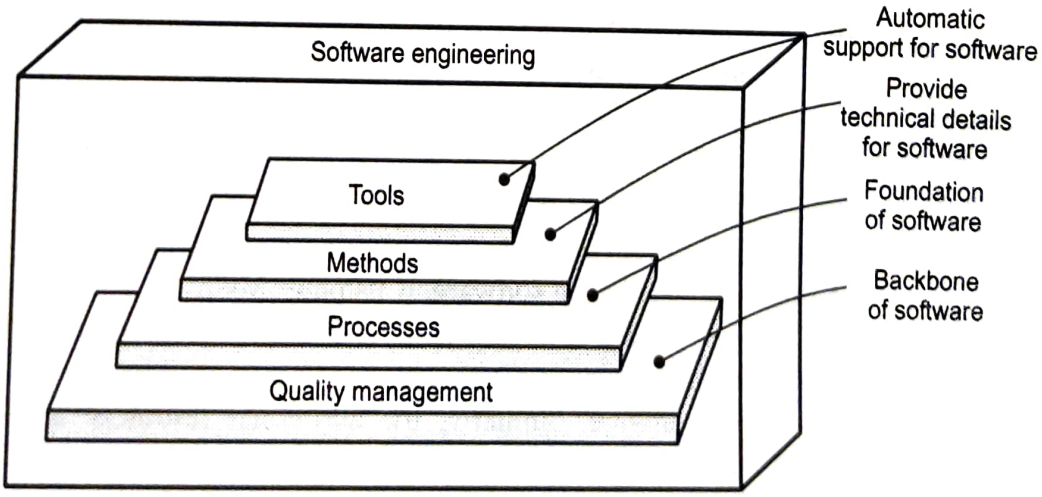

Fig. 1.7.1 Software engineering : A layered approach

- A disciplined **quality management** is a backbone of software engineering technology.

- **Process layer** is a foundation of software engineering. Basically, process defines the framework for timely delivery of software.
- In **method layer** the actual method of implementation is carried out with the help of requirement analysis, designing, coding using desired programming constructs and testing.
- Software **tools** are used to bring automation in software development process.
- Thus software engineering is a **combination** of process, methods, and tools for development of quality software.

Review Questions

1. *Define Software Engineering. Draw and explain software Engineering layers.*
   GTU : Summer-2012, Winter-2014, Marks 7
2. *Explain software engineering as a layered technology.*
   GTU : Winter-2011, Marks 3, Winter-2017,2018, Marks 3, Summer-2019, Marks 4

1.8 Process Framework

GTU : Winter-2013, 2014, Marks 7, Summer-2018, Marks 3

Software process can be defined as the structured set of activities that are required to develop the software system.

The fundamental activities are :

- Specification
- Design and implementation
- Validation
- Evolution

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

1.8.1 Common Process Framework

The process framework is required for representing the common process activities.

As shown in Fig. 1.8.1, the software process is characterized by process framework activities, task sets and umbrella activities. (See Fig. 1.8.1 on next page.)

Process framework activities

- Communication
  - By communicating customer requirement gathering is done.
- Planning - Establishes engineering work plan, describes technical risks, lists resource requirements, work products produced and defines work schedule.

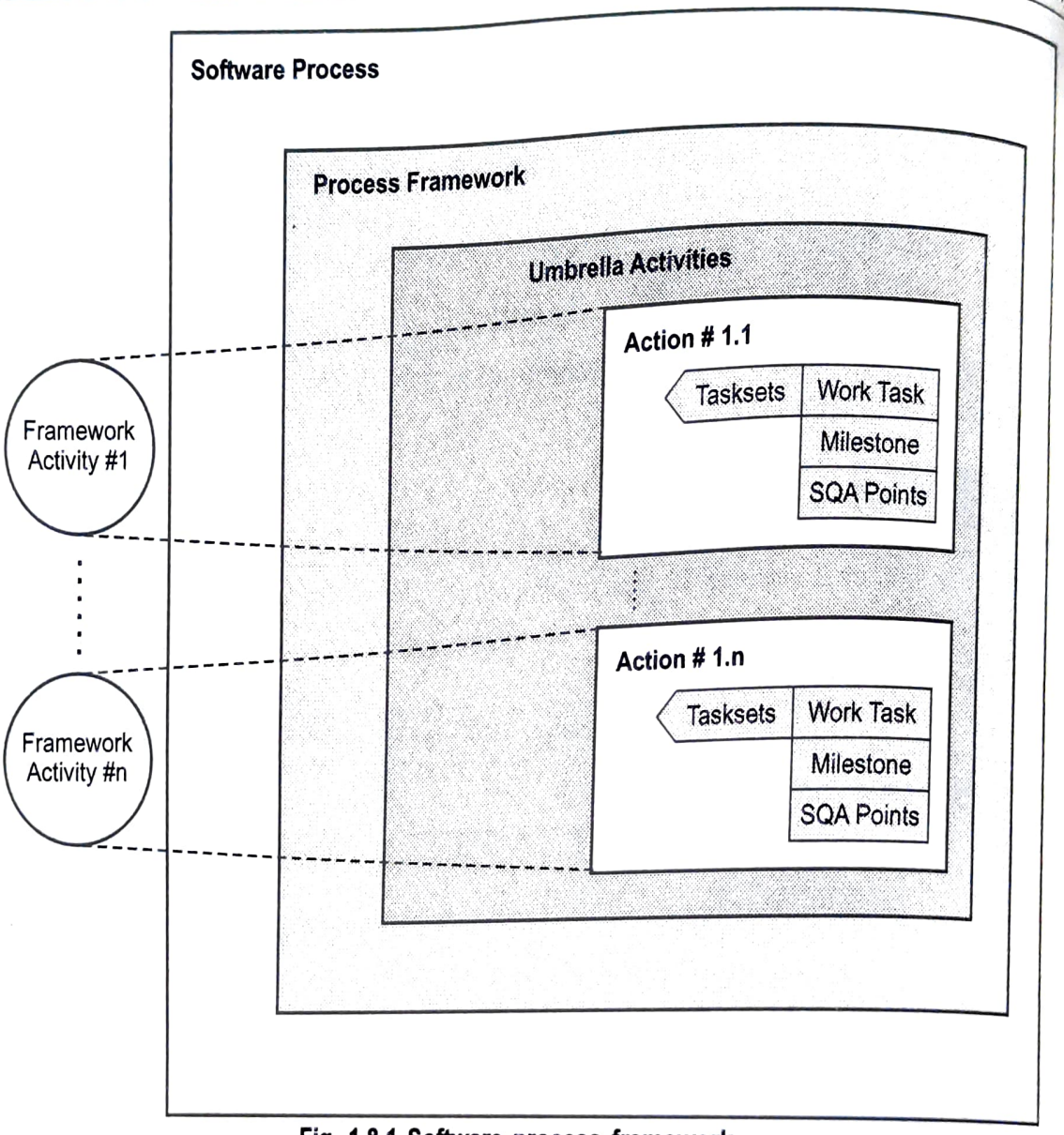

Fig. 1.8.1 Software process framework

- Modeling - The software model is prepared by :
  - Analysis of requirements
  - Design
- Construction - The software design is mapped into a code by :
  - Code generation
  - Testing
- Deployment - The software delivered for customer evaluation and feedback is obtained.

Task sets - The task set defines the actual work done in order to achieve the software objective. The task set is used to adopt the framework activities and project team requirements using :

- Collection of software engineering work tasks
- Project milestones
- Software quality assurance points

Umbrella activities - The umbrella activities occur **throughout the process**. They focus on project management, tracking and control. The umbrella activities are

1. **Software project tracking and control** - This is an activity in which software team can **assess progress** and take corrective action to **maintain schedule**.
2. **Risk management** - The **risks** that may affect project outcomes or quality can be **analyzed**.
3. **Software quality assurance** - These are activities required to **maintain** software **quality**.
4. **Formal technical reviews** - It is required to **assess engineering work products** to uncover and **remove errors** before they propagate to next activity.
5. **Software configuration management** - Managing of configuration process when any **change** in the software occurs.
6. **Work product preparation and production** - The activities to create **models, documents, logs, forms** and **lists** are carried out.
7. **Reusability management** - It defines criteria for **work product reuse**.
8. **Measurement** - In this activity, the process can be defined and collected. Also **project and product measures** are used to assist the software team in delivering the required software.

Review Questions

1. *Discuss all generic framework activities with respect to any one process model.* **GTU : Winter-2013, Winter-2014, Marks 7**
2. *Discuss umbrella activities and its role in software development life cycle (SDLC).* **GTU : Winter-2014, Marks 7**
3. *What is process ? Discuss the process framework activities.* **GTU : Summer 18, Marks 3**

1.9 Need for Software Process Model

GTU : Summer-2016, Marks 3

The software development team must decide the process model that is to be used for software product development and then the entire team must adhere to it. This is necessary because the software product development can then be done **systematically**. Each team member will **understand - what is the next activity** and **how** to do it. Thus **process model** will bring the **definiteness and discipline** in overall development process.

Every process model consists of **definite entry and exit criteria** for **each phase**. Hence the transition of the product through various phases is definite. If the process model is not followed for software development then any team member can perform

any software development activity, this will ultimately cause a chaos and software project will definitely fail. Without using process model, it is difficult to monitor the progress of software product. Let us discuss various process models one by one.

Review Question

1. *What is the importance of process model in development of software system.*

GTU : Summer-2016, Marks 3

1.10 Software Process Model

GTU : Winter-2011, 2012, 2014, 2017, 2018, 2019, Summer-2011, 2012, 2013, 2014, 2015, 2016, 2019, Marks 7

- **Definition of Process Model :** The process model can be defined as the **abstract representation of process**. The appropriate process model can be chosen based on abstract representation of process.
- The software process model is also known as **Software Development Life Cycle (SDLC) Model** or software paradigm.
- These models are called **prescriptive process models** because they are following some rules for correct usage.
- In this model various activities are carried out in some specific sequence to make the desired software product.

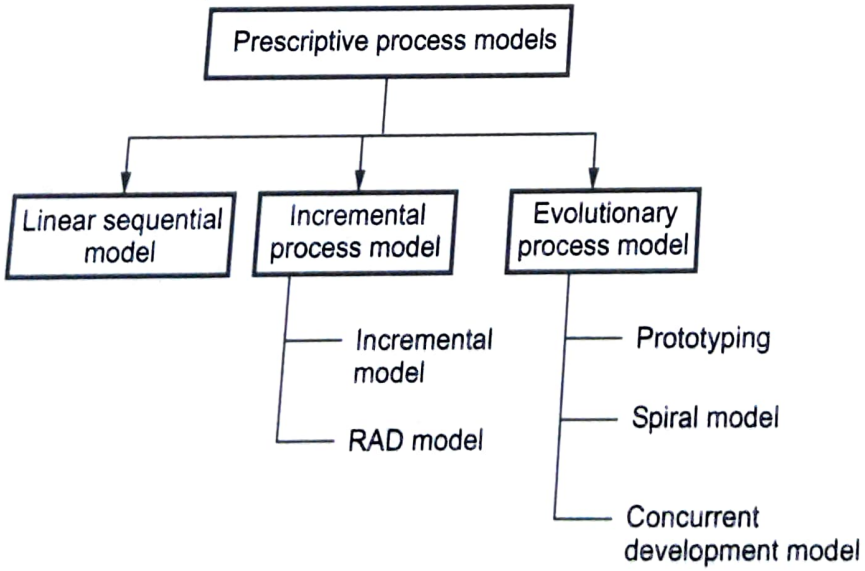

Fig. 1.10.1 Prescriptive process model

- Various prescriptive process models are as shown in Fig. 1.10.1.

1.10.1 Linear Sequential Model

- The linear sequential model is also called as **'waterfall model'** or **'classic life cycle model'**. It is the oldest software paradigm. This model suggests a systematic, **sequential approach** to software development.

- The software development starts with **requirements gathering phase**. Then progresses through **analysis, design, coding, testing** and **maintenance**. Following figure illustrates waterfall model.

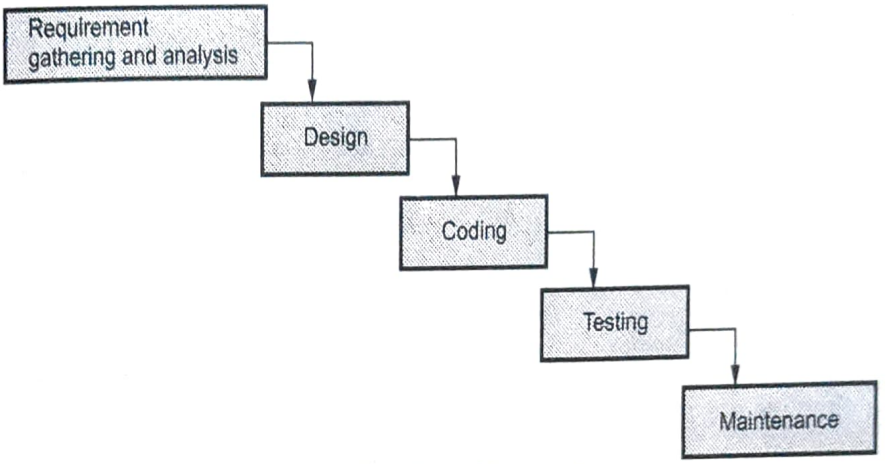

Fig. 1.10.2 Waterfall model

- In **requirement gathering and analysis** phase the **basic requirements** of the system must be understood by software engineer, who is also called **Analyst**. The **information domain, function, behavioural requirements** of the system are understood. All these requirements are then **well documented** and discussed further with the customer, for reviewing.
- The **design** is an intermediate **step between** requirements analysis and coding. Design focuses on program attributes such as -
  - Data structure
  - Software architecture
  - Interface representation
  - Algorithmic details.

  The requirements are translated in some easy to represent form using which coding can be done effectively and efficiently. The design needs to be documented for further use.
- **Coding** is a step in which design is translated into **machine-readable form**. If design is done in sufficient detail then coding can be done effectively. **Programs** are created in this phase.
- **Testing** begins when coding is done. While performing testing the major focus is on **logical internals of the software**. The testing ensures execution of all the paths, functional behaviours. The purpose of testing is to **uncover errors, fix the bugs** and **meet the customer requirements.**
- **Maintenance** is the **longest life cycle phase**. When the system is installed and put in practical use then error may get introduced, correcting such errors and **putting it in use** is the **major purpose** of maintenance activity. Similarly, enhancing **system's services** as new requirements are discovered is again maintenance of the system.

This model is widely used model, although it has many drawbacks. Let us discuss benefits and drawbacks.

Benefits of waterfall model

1. The waterfall model is **simple to implement**.
2. For implementation of **small systems** waterfall model is useful.

Drawbacks of waterfall model

There are some **problems** that are encountered if we apply the **waterfall model** and those are :

1. It is difficult to follow the sequential flow in software development process. If some changes are made at some phases then it may cause some confusion.
2. The requirement analysis is done initially and sometimes it is not possible to state all the requirements explicitly in the beginning. This causes difficulty in the project.
3. The customer can see the **working model** of the project only **at the end**. After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.
4. Linear nature of waterfall model induces **blocking states**, because certain tasks may be dependant on some previous tasks. Hence it is necessary to accomplish all the dependant tasks first. It may cause long waiting time.

Example 1.10.1 *Explain how water-fall model is applicable for the development of the following systems:*

a) University accounting system

b) Interactive system that allows railway passengers to find time and other information from the terminals installed in the station.

Solution :

a) University accounting system : If the software developers who have the experience in developing the account systems then building university account system based on existing design could be easily managed with water-fall model.

b) Interactive system that allows railway passengers to find time and other information from the terminals installed in the station.

For developing such interactive system, all the requirements must be correctly identified and analyzed before the designing of the project. The requirements of end-users must be correctly and un-ambiguously understood by the developers prior to design phase. Once the requirements are well defined then using disciplined design, coding and testing phases the required system can be built using water-fall model.

Example 1.10.2 *What is meant by 'blocking states' in linear sequential model ?*

Solution : The linear nature of linear sequential model brings a situation in the project that some project team members have to wait for other members of the team to complete the dependent tasks. This situation is called "blocking state" in linear sequential model. For example, after performing the requirement gathering and analysis step the design process can be started. Hence the team working on design stage has to wait for gathering of all the necessary requirements. Similarly the programmers can not start coding step unless and until the design of the project is completed.

Example 1.10.3 *Provide three examples of software projects that would be amendable to the waterfall model, Be specific.* **GTU : Winter-2019, Marks 3**

Solution : a) University accounting system : If the software developers who have the experience in developing the account systems then building university account system based on existing design could be easily managed with water-fall model.

b) Interactive system that allows railway passengers to find time and other information from the terminals installed in the station.

For developing such interactive system, all the requirements must be correctly identified and analyzed before the designing of the project. The requirements of end-users must be correctly and un-ambiguously understood by the developers prior to design phase. Once the requirements are well defined then using disciplined design, coding and testing phases the required system can be built using water-fall model.

c) Inventory Management System : The waterfall model is a linear sequential model in which progress can be seen as flowing steadily downwards through various phases. Once the customer is satisfied with all the requirement analysis, the further development of the project becomes simplified. This model is more appropriate for handling such projects because the requirements are very limited and specific for such type of projects.

1.10.2 Evolutionary Process Model

While developing the software systems, it is often needed to make modifications in earlier development phases or the tasks sets. If the development process is linear or in a straight line (from requirements gathering to deployment) then the end product will be unrealistic. In such cases, the **iterative approach** needs to be adopted. The evolutionary process model is **iterative model**.

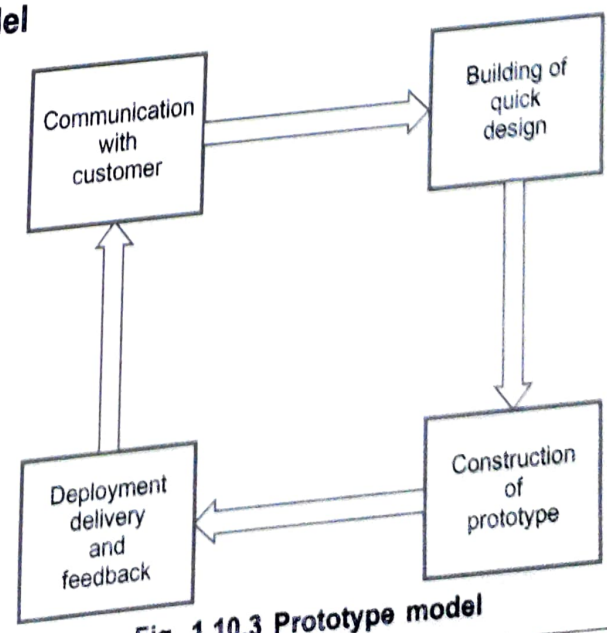

Fig. 1.10.3 Prototype model

1.10.2.1 Prototype Model

- In prototyping model initially the requirement gathering is done.
- Developer and customer define overall objectives; identify areas needing more requirement gathering.
- Then a quick design is prepared. This design represents what will be visible to user in input and output format.
- From the quick design a prototype is prepared. Customer or user evaluates the prototype in order to refine the requirements. Iteratively prototype is tuned for satisfying customer requirements. Thus prototype is important to identify the software requirements.
- When working prototype is built, developer use existing program fragments or program generators to throw away the prototype and rebuild the system to high quality.
- Certain classes of mathematical algorithms, subset of command driven systems and other applications where results can be easily examined without real time interaction can be developed using prototyping paradigm.

When to choose it ?

- Software applications that are relatively easy to prototype almost always involve Human-machine Interaction (HCI) the prototyping model is suggested.
- A general objective of software is defined but not detailed input, processing or output requirements. Then in such a case prototyping model is useful.
- When the developer is unsure of the efficiency of an algorithm or the adaptability of an operating system then prototype serves as a better choice.

Drawbacks of prototype model

1. In the first version itself, customer often wants "few fixes" rather than rebuilding of the system whereas rebuilding of new system maintains high level of quality.
2. The first version may have some compromises.
3. Sometimes developer may make implementation compromises to get prototype working quickly. Later on developer may become comfortable with compromises and forget why they are inappropriate.

Comparison between prototype model and incremental process model

Sr. No.	Prototype model	Incremental process model
1.	Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer.	The requirements are precisely defined and there is no confusion about the final product of the software.
2.	The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands.	The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase can not be tolerated.
3.	All the end-users are involved in all phases of development.	All the end-users need not be involved in all the phases of development.
4.	There can be use of some reusable software components in project development process.	There is no use of reusable components in development process.

1.10.2.2 Spiral Model

- This model possess the **iterative nature** of prototyping model and controlled and **systematic approaches** of the linear sequential model.
- This model gives efficient development of incremental versions of software. In this model, the software is developed in series of increments.
- The sprial model is divided into a number of **framework activities**. These framework activities are denoted by **task regions**.
- Usually there are **six tasks regions**. The spiral model is as shown in Fig. 1.10.4.
- Spiral model is realistic approach to development of **large-scale systems** and software. Because customer and developer better understand the problem statement at each evolutionary level. Also risks can be identified or rectified at each such level.
- In the initial pass, product specification is built and in subsequent passes around the spiral the prototype gets developed and then more improved versions of software gets developed.
- During planning phase, the cost and schedule of software can be planned and adjusted based on feedback obtained from customer evaluation.
- In spiral model, project entry point axis is defined. This axis represents starting point for different types of projects.

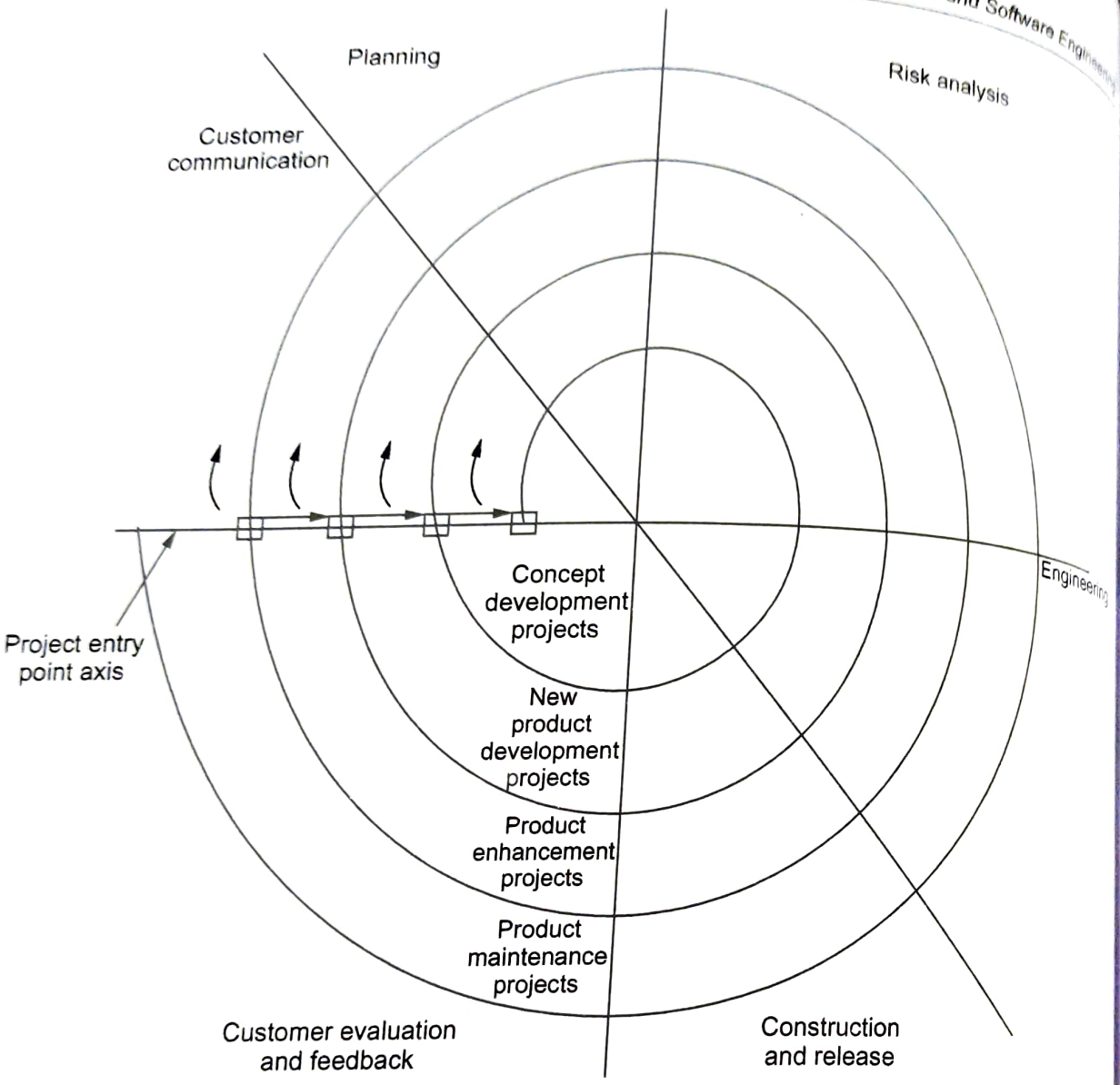

Fig. 1.10.4 Spiral model

For instance, concept development project will start at core of spiral and will continue along the spiral path. If the concept has to be developed into actual project then at entry point 2 the product development process starts. Hence entry point 2 is called product development project entry point. The development of the project can be carried out in iterations.

- The task regions can be described as :

i) **Customer communication** - In this region, it is suggested to establish customer communication.

ii) **Planning** - All planning activities are carried out in order to define resources time line and other project related activities.

iii) **Risk analysis** - The tasks required to calculate technical and management risks are carried out.

iv) Engineering - In this task region, tasks required to build one or more representations of applications are carried out.

v) **Construct and release** - All the necessary tasks required to construct, test, install the application are conducted. Some tasks that are required to provide user support are also carried out in this task region.

vi) **Customer evaluation** - Customer's feedback is obtained and based on customer evaluation required tasks are performed and implemented at installation stage.

- In each region, number of **work tasks** are carried out depending upon the characteristics of project. For a small project relatively small number of work tasks are adopted but for a complex project large number of work tasks can be carried out.
- In spiral model, the software engineering team **moves around the spiral** in a clockwise direction beginning at the core.

Advantages of spiral model

- Requirement changes can be made at every stage.
- Risks can be identified and rectified before they get problematic.

Drawbacks of spiral model

- It is based on **customer communication**. If the communication is not proper then the software product that gets developed will not be up to the mark.
- It demands considerable **risk assessment**. If the risk assessment is done properly then only the successful product can be obtained.

When to choose it ?

1. When the prototypes for the software functionality are needed.
2. When requirements are **not very clearly defined** or complex.
3. When the **large or high budget** projects need to be developed.
4. When the **risk assessment** is very critical and essential
5. When project is not expected within a specific limited time span.

Comparison between Spiral model and Prototyping model

Sr. No.	Spiral model	Prototyping model
1.	The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase can not be tolerated.	The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands.

2.	All the end-users need not be involved in all the phases of development.	All the end-users are involved in all phases of development.
3.	Funding are not stable for the projects that can be developed using spiral model.	Funding are stable for these type of projets.
4.	The requirements that are gathered and analyzed are high reliability requirements.	Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer.

Example 1.10.4 *As you move outward along with process flow path of the spiral model, what can we say about the software that is being developed or maintained ?*

Solution : When software engineering team moves around the spiral, the first circuit around the spiral results in development of product specification. The subsequent passes around the spiral might be used to develop prototype in more subsequent manner. In each pass, through planning region, some adjustments to project plan are made. Cost and schedule adjustments can also be made according to customer feedback.

Example 1.10.5 *How does "Project Risk" factor affect the spiral model of software development ?*

Solution : The spiral model demands considerable risk assessment because if a major risk is not uncovered and managed, problems will occur in the project and then it will not be acceptable by end user.

Example 1.10.6 *How does a spiral model represent a process suitable to represent a real time problem ?*

Solution : Spiral model represents a process suitable to represent a real time proble because of following reasons -

1. Software evolves as the project progresses. And at every evolutionary level t risks are identified and managed and risks are reduced at every stage.
2. It enables the developer to apply the prototype approach at any stage in the evolution of the product. It helps in adopting the approach systematic stepwise development of the product.
3. The iterative frameworks help in analyzing the product at every evolutionary stage.
4. The spiral model demands a direct consideration of technical risks at all stages of project. The risks are reduced before they get problematic.

1.10.3 Incremental Process Model

In this model with limited functionality the basic software product is created for user's understanding. This model is refined and expanded in later phases.

1.10.3.1 RAD Model

- The RAD Model is a type of incremental process model in which there is **extremely short development cycle.**
- When the requirements are fully understood and the component based **construction** approach is adopted then the RAD model is used.
- Using the RAD model the fully functional system can be developed within 60 to **90 days.**
- Various phases in RAD are Requirements Gathering, Analysis and Planning, Design, Build or Construction and finally Deployment.
- Multiple teams work on developing the software system using RAD model parallely.
- In the **requirements gathering** phase the developers communicate with the users of the system and understand the business process and requirements of the software system.

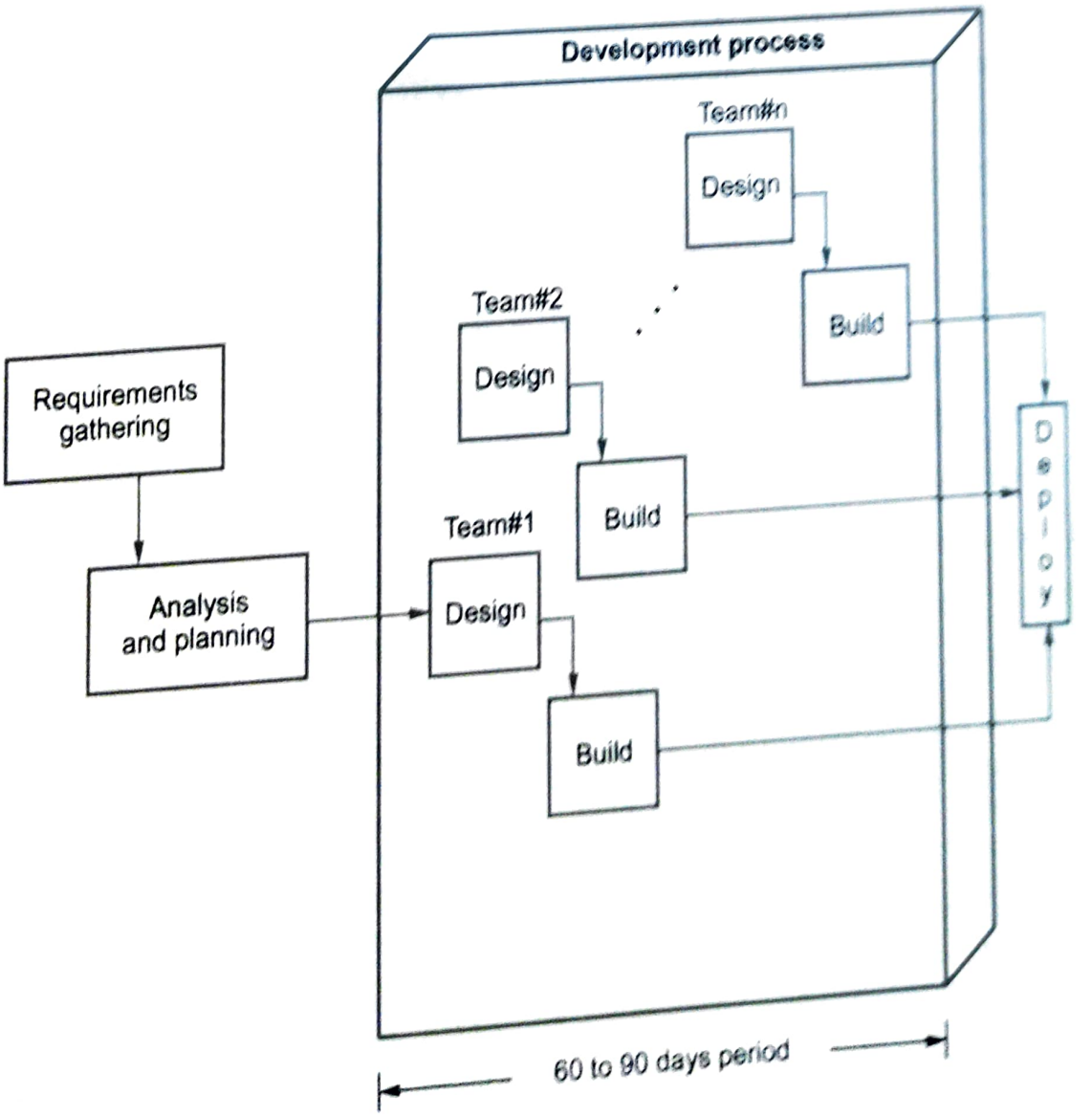

Fig. 1.10.5 Rapid application development

- During **analysis and planning** phase, the analysis on the gathered requirements is made and a planning for various software development activities is done.
- During the **design** phase various models are created. Those models are Business model, data model and process model.
- The **build** is an activity in which using the existing software components and automatic code generation tool the implementation code is created for the software system. This code is well tested by its team. The functionalities developed by all the teams are integrated to form a whole.
- Finally the deployment of all the software components (created by various teams working on the project) is carried out.

Drawbacks of rapid application development

1. It requires **multiple teams** or large number of people to work on the scalable projects.
2. This model requires **heavily committed developer** and customers. If commitment is lacking then RAD projects will fail.
3. The projects using RAD model requires **heavy resources.**
4. If there is no appropriate modularization then RAD projects fail. Performance can be problem to such projects.
5. The projects using RAD model find it difficult to adopt new technologies.

Example 1.10.7 *Which type of applications suit RAD model ? Justify your answer.*

Solution : The RAD model is suitable for information system applications, business applications and the for systems that can be modularized because of following reasons -

1. This model is similar to waterfall model but it uses very short development cycle.
2. It uses component-based construction and emphasises reuse and code generation.
3. This model uses multiple teams on scaleable projects.
4. The RAD model is suitable for the projects where technical risks are not high.
5. The RAD model requires heavy resources.

Example 1.10.8 *Provide three examples of software projects that would be amenable to incremental model. Be specific.*

Solution : There can various examples of software projects that would be amenable to incremental model. For instance -

1. Banking software service : This service can be personal service. That means for personal banking system the incremental model can be used. In later state of increments, this system can implement insurance service, home loans and some other features of banking services.

2. Web browser application : The base application can be developed and distributed. This is the basic increment of the application. In the later increments the plugins can be provided to enhance the experience of web browser applications.
3. Operating system software : The operating system software providing the basic system handling functionalities is the first increment. After the release of the basic versions then updates or security patches are provided to the customer in the form of increments. Various distribution packge in the form of versions such as basic home edition, premium, ultimate and so on can be the increments of operating system software.

1.10.4 Comparison between Various Process Models

1. Evolutionary and Incremental Process Model

Sr. No.	Evolutionary process model	Incremental process model
1.	Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer.	The requirements are precisely defined and there is no confusion about the final product of the software.
2.	The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands	The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase cannot be tolerated.
3.	All the end-users are involved in all phases of development	All the end-users need not be involved in all the phases of development.
4.	There can be use of some reusable software components in project development process	There is no use of reusable components in development process.

2. Waterfall Model and Spiral Model

Sr. No.	Waterfall model	Spiral model
1.	It requires well **understanding of requirements** and familiar technology.	It is developed in **iterations**. Hence the requirements can be identified at new iterations.
2.	**Difficult** to accommodate **changes** after the process has started.	The required changes can be made at every stage of **new version**.
3.	Can accommodate iteration but indirectly.	It is iterative model.
4.	**Risks** can be identified **at the end** which may cause failure to the product.	**Risks** can be identified and **reduced** before they get problematic.

5.	The customer can see the **working model** of the project only **at the end**. After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.	The customer can see the **working product** at certain **stages** of iterations.
6.	**Customers prefer** this model.	Developers prefer this model.
7.	This model is good for **small systems**.	This model is good for **large systems**.
8.	It has **sequential** nature.	It has **evolutionary** nature.

3. Waterfall, Spiral, Prototype and Incremental Model

Waterfall model	Spiral model	Prototyping model	Incremental model
Requirements must be clearly understood and defined at the beginning only.	The **requirements** analysis and gathering can be done in iterations because requirements get changed quite often.	**Requirements analysis** can be made in the later stages of the development cycle, because requirements get changed quite often.	The **requirements analysis** can be made in the later stages of the development cycle.
The **development team** having the adequate experience of working on the similar project is chosen to work on this type of process model	The **development team** having less experience of working on the similar projects is allowed in this process model	The **development team** having less experience of working on the similar projects is allowed in this process model.	The **development team** having the adequate experience of working on the similar project is chosen to work on this type of process model.
There is **no user involvement** in all the phases of development process.	There is **no user involvement** in all the phases of development process.	There is **user involvement** in all the phases of development process.	There is **user involvement** in all the phases of development process.
When the requirements are reasonably well defined and the development effort suggests a purely linear effort then the **waterfall model is chosen.**	Due to iterative nature of this model, the risk identification and rectification is done before they get problematic. Hence for handling real time problems the **spiral model is chosen.**	When developer is unsure about the efficiency of an algorithm or the adaptability of an operating system then the **prototyping model is chosen.**	When the requirements are reasonably well defined, the development effort suggests a purely linear effort and when limited set of software functionality is needed quickly then the incremental **model is chosen.**

4. Prototype Model and RAD Model

Sr. No.	Prototype Model	RAD Model
1.	Requirements are gathered initially but there can be change in the requirements in the later stage.	The RAD model is basically an incremental model and requirements are precisely defined and there is no change in the requirements.
2.	All the requirements can't be specified in the early stage of the model.	All the requirements are defined in the early stage of the model.
3.	This model allows the development team that have less experience on similar projects.	This model require that the development team must be well experienced.
4.	The project can be developed with limited number of resources using prototype model.	The RAD model requires heavy number of resources.
5.	The training for the development team is not available for such type of projects.	The people can be trained to work on projects using RAD model.
6.	Requirements can indicated the complexity of the projects.	The requirements can not indicate the complexity of the projects.
7.	Users of the projects that are developed using Prototype model can be novice or less experiences persons.	Users of the projects that are developed using RAD model are experienced persons. They are the persons who have used previously used such type of projects.

Review Questions

1. *Explain in brief the spiral model.* **GTU : Summer-2011, Winter-2011, Marks 7**
2. *Explain in brief the process model which is used in situations where requirements are well defined and stable. (**Hint** : Waterfall model - Refer section 1.10.1).* **GTU : Summer-2011, Marks 7**
3. *Explain incremental model for system development. Differentiate it with spiral model* **GTU : Summer-2012, Marks 7**
4. *Explain spiral model and its advantages. Compare prototype model and spiral model.* **GTU : Winter-2012, Summer-2014, Marks 7**
5. *Explain in detail the process model which is normally suited for the development of large scale software system. (**Hint** : spiral model - Refer section 1.10.2.2)* **GTU : Summer-2013, Marks 7**
6. *Describe two main features of spiral model and discuss working of prototyping model with its diagram.* **GTU : Winter-2014, Marks 7**
7. *Discuss incremental process model with its diagram and compare with waterfall model.* **GTU : Winter-2014, Summer-2015, Marks 7**

8. *Explain prototype model and compare it with waterfall process model.* **GTU : Summer-2014, Marks 7**
9. *Compare prototype and RAD model.* **GTU : Summer-2015, 2019, Winter-2017, 2018, Marks 3**
10. *Explain prototype process model.* **GTU : Summer-2016, Marks 4**
11. *Explain the process model which is used for development of large-scale system.* **GTU : Summer-2016, Marks 7**
12. *Explain the process model which is normally suits for development of large - scale software system* **GTU : Winter-2017, Marks 4**
13. *Explain spiral model in brief with suitable diagram.* **GTU : Winter-2018, Marks 4**
14. *What is the importance of process model in development of software system ? Explain prototype process model.* **GTU : Winter-2018, Marks 7**
15. *Explain Waterfall process model.* **GTU : Summer-19, Winter-2019, Marks 7**

1.11 Agile Process Models

GTU : Winter-2017, Marks 4

The agile processes are the light-weight methods are **people-based** rather than plan-based methods. The agile process forces the development team to **focus** on **software** itself rather than design and documentation. The agile process believes in **iterative method.** The aim of agile process is to **deliver** the working model of software **quickly** to the customer.

There are various process models used as agile process model. But the most famous agile process model is extreme Programming.

Review Question

1. *Explain agile development in detail.* **GTU : Winter 17, Marks 4**

1.12 Component Based Development

- The commercial off-the-shelves components that are developed by the vendors are used during the software built.
- These components have specialized targeted functionalities and well defined interfaces. Hence it is easy to integrate these components into the existing software.
- The component based development model makes use of various characteristics of **spiral model.** This model is evolutionary in nature. That means the necessary changes can be made in the software during the each iteration of software development cycle.

- Before beginning the modelling and construction activity of software development the **candidate component** must be searched and analyzed. The components can be simple functions or can be object oriented classes or methods.
- Following steps are applied for component based development -
  - Identify the component based products and analyze them for fitting in the existing application domain.
  - Analyze the component integration issues.
  - Design the software architecture to accommodate the components
  - Integrate the components into the software architecture.
  - Conduct comprehensive testing for the developed software.
- **Software reusability** is the major advantage of component based development.
- The **reusability** reduces the development cycle time and overall cost.

1.13 Product and Process

GTU : Winter-2011, Summer-2014, 2019, Winter-2019, Marks 7

Development of product depends upon the process which is followed during the development.

The comparison between the two is as follows -

Sr. No.	Software process	Software product
1.	Processes are developed by individual user and it is used for personal use.	Software product is developed by multiple users and it is used by large number of people or customers.
2.	Process may be small in size and possessing limited functionality. It defines framework for effective product generation.	Software product consists of multiple program codes, related documents such as SRS, designing documents user manuals, test cases and so on.
3.	Generally only one person uses the process, hence there is a lack of user interface.	Good graphical user interface is most required by any software product.
4.	Process is generally developed by process engineers.	Software product is developed by **software engineers** who are large in number and work in a team. Therefore systematic approach of developing software product must be applied.

5.	The output of process is product.	Product development occurs by following process.
6.	**For example :** Program developed for parsing the input.	**For example :** A word processing software.

Review Questions

1. *Distinguish between a program and software product.* **GTU : Winter-2011, Marks 4**
2. *What is software engineering ? What is process ? What is product ?* **GTU : Summer-2014, Marks 7**
3. *What is software engineering ? What is process ? What is product ?* **GTU : Summer-2019, Marks 3**
4. *Compare product and process.* **GTU : Winter-2019, Marks 4**