

Introduction to Software Engineering (4340702,C403)

Course Outcomes

Course Outcomes (COs)	Statement
C403.1	Able to Compare various software development process models
C403.2	Able to Prepare software analysis and design using SRS, DFD and object oriented UML diagrams
C403.3	Able to Prepare software development plan using project scheduling.
C403.4	Able to Prepare test-cases to test software functionalities.

SUGGESTED SPECIFICATION TABLE WITH HOURS & MARKS (THEORY)

Unit	Unit Title	Teaching Hours	Distribution of Theory marks			
			R Level	U Level	A Level	Total
1	Software Development Process Models	10	06	08	04	18
2	Software Requirement Analysis and Design	14	04	08	08	20
3	Software Project Estimation and scheduling	10	06	08	04	18
4	Software Coding and testing	08	04	06	04	14
Total		42	22	32	16	70

TEXT/REFERENCE BOOKS

Sr. No.	Title of Books	Author	Publication
1.	Software Engineering: A Practitioner's Approach	Roger S. Pressman	Tata McGraw Hill,
2.	Software Engineering	Ian Somerville	Pearson education PHI
3.	Fundamentals of Software Engineering	Rajib Mall	PHI
4.	Structured System analysis and Design	Madhulika JAin	Bph Publication
5.	Object Oriented Modeling and design with UML, second edition	Michael R Blaha and James R Rumbaugh	Pearson Prentice Hall

Unit : 1

Software

Development Process

Teaching Hours	Distribution of Theory marks			
	R Level	U Level	A Level	Total
10	06	08	04	18

1.1 Software

❖ Definition:

- **(IEEE Definition):** Software is the “Collection of computer programs, procedures, rules, associated documents and concerned data with the operation of data processing system.”
- It also includes representations of pictorial, video, and audio information.

Software Engineering

❖ Various Definition:

- SE is an engineering discipline that covers all aspects of s/w from specification to maintenance.
- SE is an engineering discipline that delivers high quality s/w at agreed cost & in planed schedule.
- SE provide framework that guides the s/w engineers to develop the software.
- SE covers technical and management issues.
- Three main aspects of SE is → (Quality S/W at agreed cost in schedule time)
 - Provide quality product
 - Expected cost
 - Complete work on agreed schedule

Software Engineering Definition

- SE is the establishment and use of sound engineering principles in order to obtain economically s/w that is reliable and work efficiently on real machines.
- **(IEEE Definition)** → “Software engineering is the application of a symmetric , disciplined and quantifiable approach to the development, operation and maintenance of software.”
- **(Sommerville):** Software Engineering is concerned with the theories, methods and tools to develop the software products in a cost effective way.



Software Application Domain

- Refers to specific area of computer application in which it is designed to operate
- Differs in language, platform, data storage and retrieval, UI, technical specification
- **Software types:**
 - *System software*: it is responsible for controlling, integrating the hardware components of a system so the software and the users can work with them.
Example: Operating system
 - *Application software*: it is used to accomplish some specific task. It should be collection of small programs.
Example: Microsoft word, Excel, Railway reservation system.
- Software is logical rather than physical.



Software Application Domain

- **System s/w:**
 - Interface between h/w and s/w.
 - Designed to manage and control h/w & other s/w
 - Controls peripherals.
 - Ex:OS,Device drivers, language translators etc

Software Application Domain

- Application s/w:
 - Designed to serve specific need of user by performing specific task.
 - Directly accessed by user.
 - provides user friendly interface to interact with like browser ,document etc
 - Ex: word processing, excel,browser, media players, photo editor , Computer-aided design (CAD), System simulation software etc

Software Application Domain

- **Embedded s/w:**
 - Embedded in hardware of computer, peripherals and handheld device product
 - Operates in Real time environment.
 - Specifically interacts with hardware.
 - Ex: car automation, image processing system in cameras, navigation system in vehicles, controller(ROM) of washing machines, microwaves, etc
 - Limitations: high cost, complexities, limited upgradation etc.

Software Application Domain

- **Web applications:**
 - Runs in remote servers
 - Implemented using ASP.net ,html,css,JS,react JS,NODE js,etc. web technologies
 - Implement, RMI,RPC Distributed DB,OS etc.
 - Characteristic: Availability, Client driven, responsive, Customizable,Transaction oriented, Security, Large data handelling, speedy performance
 - Ex.: e-commerce ,online banking, LMS,Google Docs,Cloud based applications etc.

Software Application Domain

- AI software:
 - Provides thinking ability to computer.
 - Needs intelligent algorithm to process large data and to learn from pattern , heuristic data.
 - Lets s/w to learn from data, identify features and patterns and predict future trends.
 - Includes: neural n/w, ML,Deep Learning,NLP, Signal processing, pattern recognition software
 - Ex:Google Assistant,SIRI,Alexa,Cortana,ChatGPT



Software Characteristics

- These are the attributes reflect the quality of a software product.
- Following are the characteristics of good software. (Qualities for good software).
 - ↳ *Understandability*
 - ↳ *Cost*
 - ↳ *Maintainability*
 - ↳ *Modularity*
 - ↳ *Functionality*
 - ↳ *Reliability*
 - ↳ *Portability*
 - ↳ *Correctness*
 - ↳ *Documentation*
 - ↳ *Reusability*
 - ↳ *interoperability*

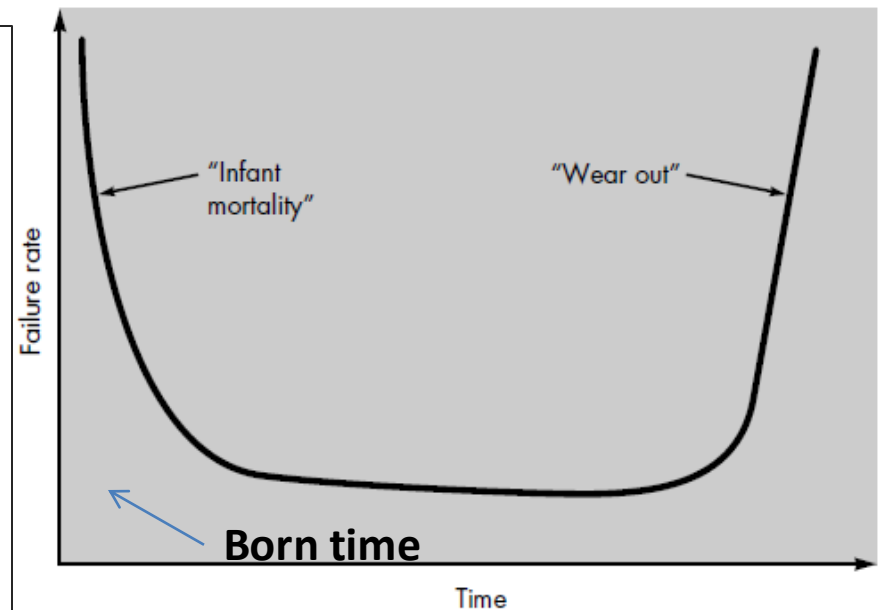
Software Characteristics

◆ Some special characteristics of s/w:

❑ Software doesn't wear out.

- Hardware can damage after running time. It can be affected by environmental effects. So the failure rate rises.

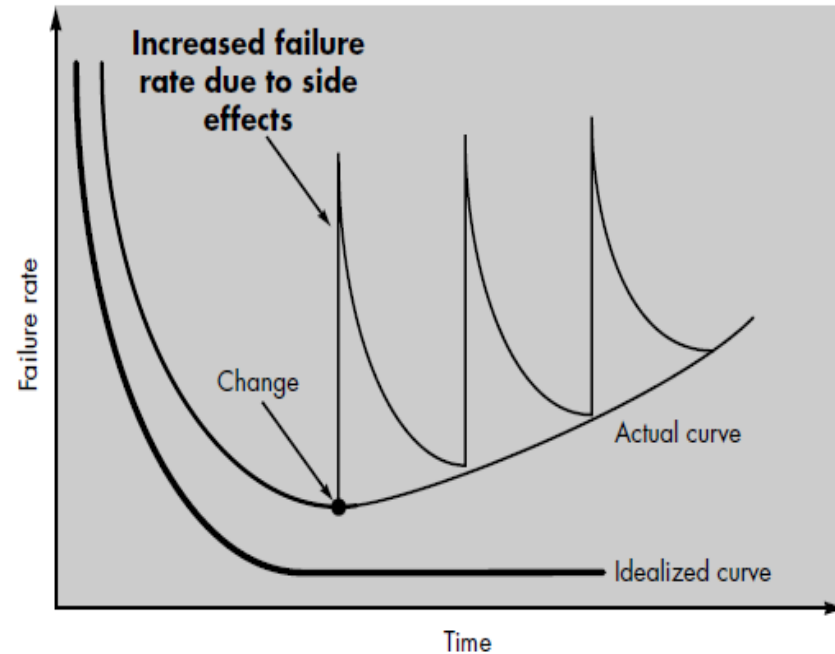
- “**bathtub curve**” shows hardware failure
- there are three phases in h/w life
 - initially failure rate is much more. But after testing and defects are corrected, failure rate come down.
 - In it, h/w is much more useful and chance of failure is quite low.
 - As time passes, however, the failure rate rises again as hardware components suffer from the affects of dust, vibration, abuse, temperature extremes, and many other environmental factors.
 - So simply, hardware does wear out.



H/W failure curve

Software Characteristics

- Software is **not highly affected by environmental effects**. The “idealized curve” shows software failure.
- In the early stage, due to lot many errors, software could have high failure. But it becomes reliable as time passes instead of wearing out. Software become reliable.
- Software may be retired due to new requirements, new expectations etc.
- **Hence, software doesn't wear out, but it may be deteriorate.**



S/W failure curve

❑ S/W is engineered, not manufactured.

Once a product is manufactured, it is not easy to modify it, change. While in case of software we can easily change or modify or change it for later use.

Even making multiple copies of software is a very easy.

Software Characteristics

In hardware, costing is due to assembly of raw material and other processing expenses while in software development no assembly needed like hardware. Hence, software is not manufactured as it is developed or it is engineered.

☐ Reusability of components.

Self description.

☐ Software is flexible for custom built.

A program can be developed to do anything. Any kind of change needed in software easily done.

A software program or product can be built on user requirements basis or custom built.

Programs versus Software Products

Program	Software product
- It is usually small in size.	- It is large in size.
- It has single developer.	- Here, team of developers.
- Author himself is sole user.	- Large number of users.
- It lacks proper user interface.	- Here, well designed interface.
- It lacks proper documentation.	- Here, well documented and user manual prepared.
- It is ad-hoc development.	- It is systematic development.
- No need of systematic methodologies.	- Require proper systematic methodologies.

Software Myths

- Software myths propagated misinformation and confusion.
- There are many myths of software and software engineering in software development community.
- **Myth: Software is easy to change.**
- Reality: ----
- **Myth: Outsourcing of s/w to a third party can relax the customers.**
- Reality: ---
- **Myth: Software can work right the first time.**
- Reality: ---
- **Myth: Increasing of software reliability will increase software safety.**
- Reality: ---

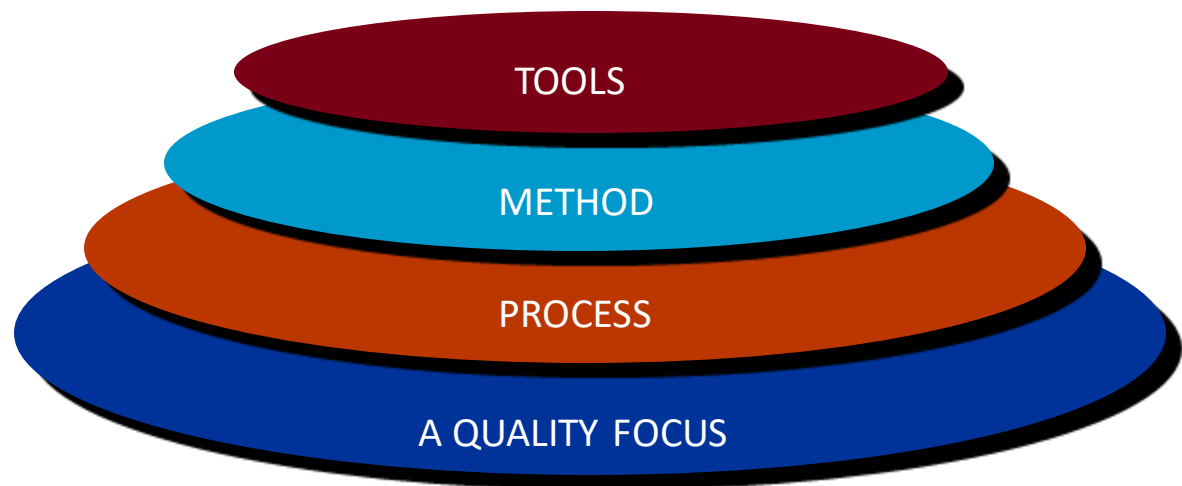
Software Myths

- **Myth: Reusing software increase safety.**
- Reality: ---
- **Myth: Best software is one which has more features.**
- Reality: ---
- **Myth: Testing of software will remove all errors.**
- Reality: ---
- **Myth: once the project is working, job is done.**
- Reality: ---



1.3 Software Engineering Layered approach

- Software engineering can be viewed as **a layered technology**. Actually software engineering is totally a layered technology.
- It **encompasses process, methods, tools that enables a s/w product to be built in a timely manner**.
- Four layers are there.
 - Quality
 - Process
 - Method
 - Tools



SE Layers

Software Engineering Layered approach

- A Quality focus Layer
 - SE mainly focuses on quality product.
 - It checks whether the **output meets with its requirement specifications** or not.
 - Every organization should maintain its total quality management.
 - This layer supports software engineering.
- Process Layer
 - It is the heart of the SE.
 - It is a foundation layer for development.
 - s/w **process is a set of activities together if ordered and performed properly, the desired result would be produced.**
 - Define framework activities.
 - Main idea → ***is to deliver s/w in a timely manner.***
- Method Layer
 - It describes '**how-to**' **build** software product.
 - It includes user interaction, requirement analysis, designing, coding, testing maintenance.
- Tools layer
 - It provides support to below layers.
 - Execute process in proper manner.

Need of Software Engineering?

- To help developers *to obtain high quality software product*.
- To develop the product *in appropriate manner* using life cycle models.
- To acquire skills *to develop large programs*.
- To acquire skills *to be a better programmer*.
- To provide a software product *in a timely manner*.
- To provide *a quality software product*.
- To provide a software product *at a agreed cost*.
- To develop ability *to solve complex programming problems*.
- Also learn techniques of: specification, design, user interface development, testing, project management, etc.

1.4 Generic Framework and Umbrella activities

Common Process Framework

Framework Activities

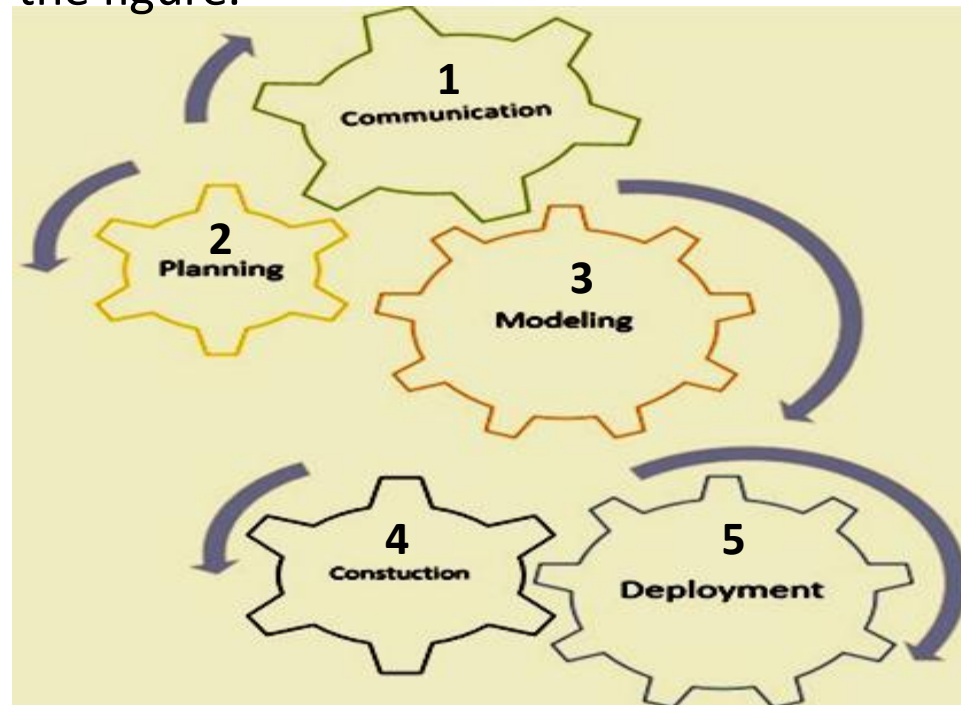
Task set

- Tasks
- Milestones, deliverables
- QA checkpoints

Umbrella Activities

1. Project tracking and control
2. Formal technical review
3. SW quality assurance
4. SW Configuration Management (SCM)
5. Document preparation and production
6. Reusability management
7. Risk management

- Each framework activity is populated by set of task, milestones and quality assurance.
- Umbrella activities are performed through out the process.
- these are independent of any framework activity.
- the list of umbrella activities are given in the figure.



Generic Framework :

1. Communication

By communication, customer **requirement gathering** is done. Communication with consumers and stakeholders to determine the system's objectives and the software's requirements.

3. Modeling

Architectural models and design to better understand the problem and to work towards the best solution. The software model is prepared by:

- **Analysis of requirements**
- **Design**

5. Deployment

In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for the supply of better products

2. Planning

Establish engineering work plan, describes **technical risk, lists resources requirements**, work produced and defines **work schedule**.

4. Construction

Creating **code, testing the system, fixing bugs**, and confirming that all criteria are met. The software design is mapped into a code by:

- **Code generation**
- **Testing**

Software process

Process framework

Umbrella activities

framework activity # 1

software engineering action #1.1

Task sets



software engineering action #1.k

Task sets

work tasks
work products
quality assurance points
project milestones

work tasks
work products
quality assurance points
project milestones



framework activity # n

software engineering action #n.1

Task sets



software engineering action #n.m

Task sets

work tasks
work products
quality assurance points
project milestones

work tasks
work products
quality assurance points
project milestones

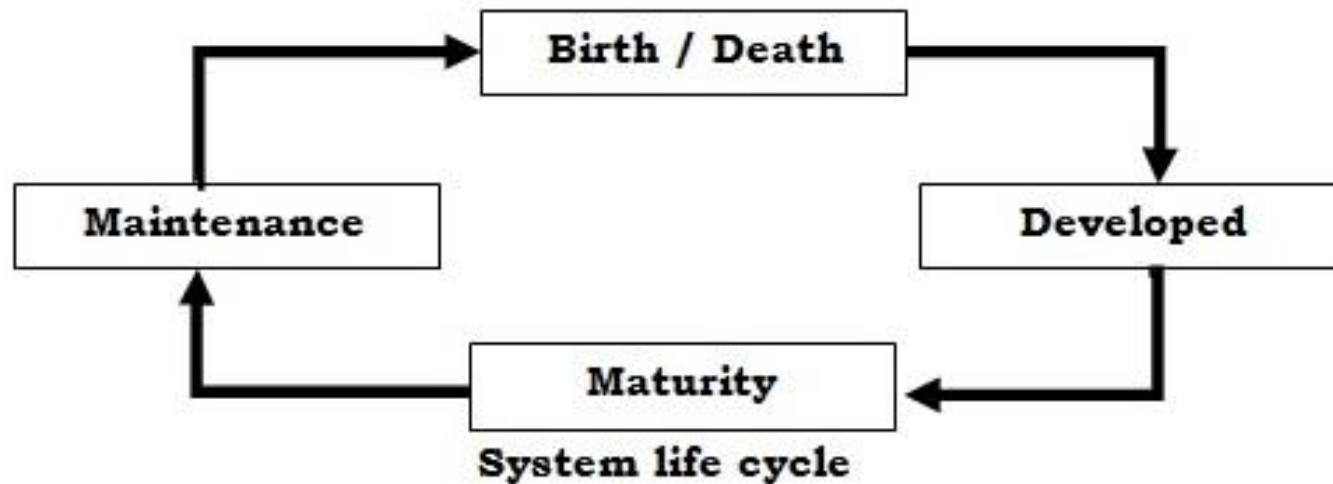
Umbrella activities

- ❑ Software project tracking and control
 - Assessing progress against the project plan.
 - Take adequate action to maintain schedule.
- ❑ Formal technical reviews
 - Assessing software work products in an effort to uncover and remove errors before goes into next action or activity.
- ❑ Software quality assurance
 - Define and conducts the activities required to ensure software quality.
- ❑ Software configuration management
 - Manages the effects of change.
- ❑ Document preparation and production
 - Help to create work products such as models, documents, logs, form and list.
- ❑ Reusability management
 - Define criteria for work product reuse
 - Mechanisms to achieve reusable components.
- ❑ Measurement
 - Define and collects process, project, and product measures
 - Assist the team in delivering software that meets customer's needs.
- ❑ Risk management
 - Assesses risks that may effect that outcome of project or quality of product (i.e. software)

1.6 Software Development Life cycle

Models

- Every system has a life cycle. It begins when a problem is recognized, after then system is developed, grows until maturity and then maintenance needed due to change in the nature of the system.



- Goal → is to produce high quality software product.
- As per IEEE Standards, software life cycle is: “the period of time that starts when software product is conceived and ends when the product is no longer available for use.”
- A software life cycle model is also called a Software Development Life Cycle (SDLC).

1.6 Software Development Life cycle

Models

- Software life cycle is the ***series of identifiable stages*** that a s/w product undergoes during its lifetime.
- Software life cycle model (process model) → is a ***descriptive and diagrammatic representation*** of the software life cycle.
- A life cycle model represents all the activities required to make a software product transit through its life cycle phases.
- ***General stages*** → feasibility study, requirement analysis and specification, design, coding, testing and maintenance.
- Every software development process model includes system requirements as ***input*** and deliverable product as ***output***.

1.6 Software Development Life cycle

Models

- **Need of life cycle models.**
- Provide *generic guidelines* for developing a suitable process for a project.
- It provides improvement and guarantee of *quality product*.
- Without using of a particular life cycle model the development of a software product would not be in a *systematic and disciplined manner*.
- Provide *monitoring the progress*.
- Defines *entry and exit criteria*.
- The *documentation* of life cycle models enhances the understanding between developers and client.
- **Different software life cycle models.**



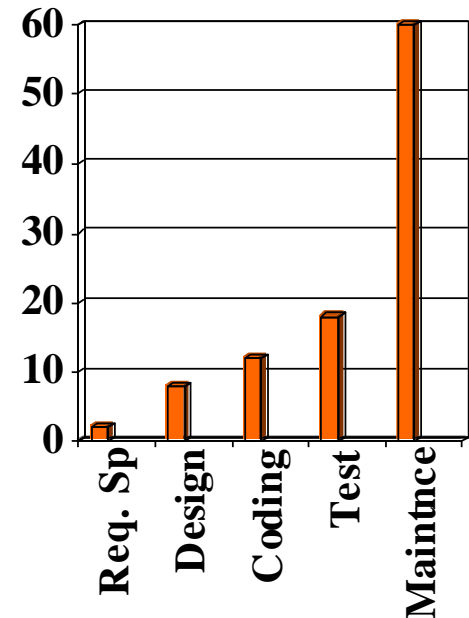
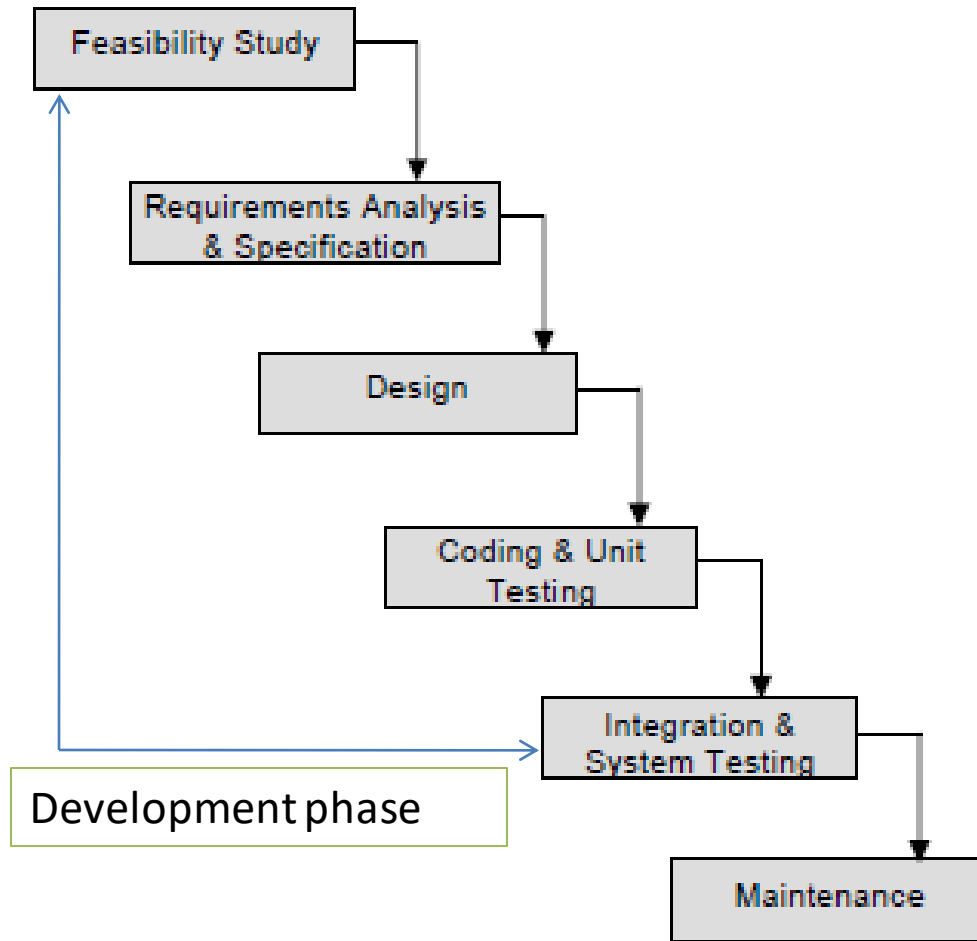
1.6.1 Classical Waterfall model

- This model was originally proposed by Royce (1970).
- It is also called 'traditional waterfall model' or 'conventional waterfall model'.
- It is the most obvious way to develop the software.
- It's just a theoretical way of developing s/w. All other life cycle models are essentially derived from it.

Phases of this model are:

- Feasibility study
- Requirements analysis and specification,
- Design
- Coding and unit testing
- Integration and system testing
- Maintenance

1.6.1 Classical waterfall model



Phases between feasibility study and testing

known as **development phases**.

Among all life cycle phases

maintenance phase consumes maximum effort.

Among development phases,

testing phase consumes the maximum effort.

1.6.1 Classical Waterfall model

◆ Feasibility Study.

- Aim → To determine whether the system would be **financially and technically feasible** to develop the product
- Includes the analysis of the problem and collection of relevant info of i/p, processing and o/p data
- Collected data are analyzed :
 - For an abstract definition
 - For Formulation of different solutions
 - For Analysis of alternative solutions
- Cost / benefit analysis is performed.
- Three main issues are concerned with feasibility:
 - Technical feasibility
 - Economical feasibility
 - Operational feasibility

1.6.1 Classical Waterfall model

◆ Requirement Analysis and Specification.

- Aim → is to **understand the exact requirements** of the customer and to document them properly.
- Also **reduces communication gap** between developers and customers.
- Two different activities are performed during this phase:
- ***Requirements gathering and analysis.***
- ***Requirements specification***

↪ **Requirement specification:**

- In it ,user **requirements are systematically organized** into a Software Requirements Specification (**SRS**) document.
- The important components of SRS are – the functional requirement, the non functional requirement and the goals of implementations.
- Output of this phase is → SRS document, which is also called *Black box specification* of the problem.

This phase concentrates on “what” part not “how”.

1.6.1 Classical Waterfall model

◆ Design.

- The goal of the design phase is to **transform the requirements** specified in the SRS document **into a structure** that is suitable for implementation in some programming language.
- This phase affecting the quality of the product.
- Two main approaches are concerned with this phase.
 - I. Traditional design approach*
 - II. Object oriented design approach*
- **Traditional design** consists of two different activities :
 - I. Structural Analysis:**
 - Where the **detailed structure** of the problem is examined.
 - **Identify the processes and data flow** among these processes.
 - **DFD is used** to perform the structure analysis and to produce result.
 - In structure analysis – functional requirement specified in SRS are decomposed and **analysis of data flow** is represented diagrammatically by **DFD**.
 - Analysis Phase involves **data flow diagram, data dictionary, state transition diagram, and entity-relationship diagram**

1.6.1 Classical Waterfall model

II. Structure design:

- During structured design, the results of structured analysis are transformed into the software design.
- Two main activities are associated with it :
 - 1. Architectural design (High-level design)**
 - decomposing the system into modules and build relationship among them. **Class diagram**
 - 2. Detailed design (Low-level design)**
 - identified individual modules are design with data structure and algorithms.

1.6.1 Classical Waterfall model

- In ***object oriented approach***, objects available in the system and relationships are identified during this approach.
- Several tools and techniques are used in designing like:
 - Flow chart
 - DFD
 - Data Dictionary
 - Decision Table
 - Decision Tree
 - Structured English

1.6.1 Classical Waterfall model

◆ Coding and unit testing.

- It is also called the **implementation phase**.
- The aim of this phase is → **to translate the software design into source code** and unit testing is done module wise.
- Each component of the design is implemented as a program module, and each unit is tested to determine the correct working of the system.
- System is being operational in this phase.
- This phase affects testing and maintenance.
- Simplicity and clarity should be maintained.
- **Output** of this phase is → **set of program** modules that have been individually tested.

1.6.1 Classical Waterfall model

◆ Integration and system testing.

- Once all the modules are coded and tested individually, integration of different modules is undertaken.
- This phase is carried out incrementally over a number of steps and during each integration step, the partially integrated system is tested and a set of previously planned modules are added to it.
- Finally, when all the modules have been successfully integrated and tested, system testing is carried out.
- Goal of this phase is → to ensure that the developed system works well to its requirements described in the SRS document.
- Testing procedure is carried out using test data: Program test and System test. Program test is done on test data and system test is done actual data.
- It consists of three different kinds of testing activities.

α – testing

β – testing

Acceptance testing

- Output of this phase is → system test plan and test report (error report).

1.6.1 Classical Waterfall model

◆ Maintenance.

- It requires max efforts to develop s/w product.
- This phase is needed to keep system operational.
- Generally maintenance is needed due to change in the environment or the requirement of the system.
- Maintenance involves performing following three kinds of activities:
 - I. Corrective maintenance**
 - II. Perfective maintenance**
 - III. Adaptive maintenance**
- In this phase the system is reviewing for full capabilities of the system.

1.6.1 Classical Waterfall model

◆ Advantages:

- It is **simple and easy** to understand and use.
- Each phase **has well defined input and outputs**.
- It helps **project personnel in planning**.
- Waterfall model **works well for smaller projects** where **requirements are very well understood**.
- It divides complex tasks into smaller, more **manageable works**.

◆ Disadvantages:

- It is a **theoretical model**.
- It may happen that the **error may be generated at any phase** and **encountered in later phase**. So it's not possible to go back and solve the error in this model.
- **High amounts of risk** and uncertainty.
- **Formal documents** are needed at each phase.

1.6.1 Classical Waterfall model

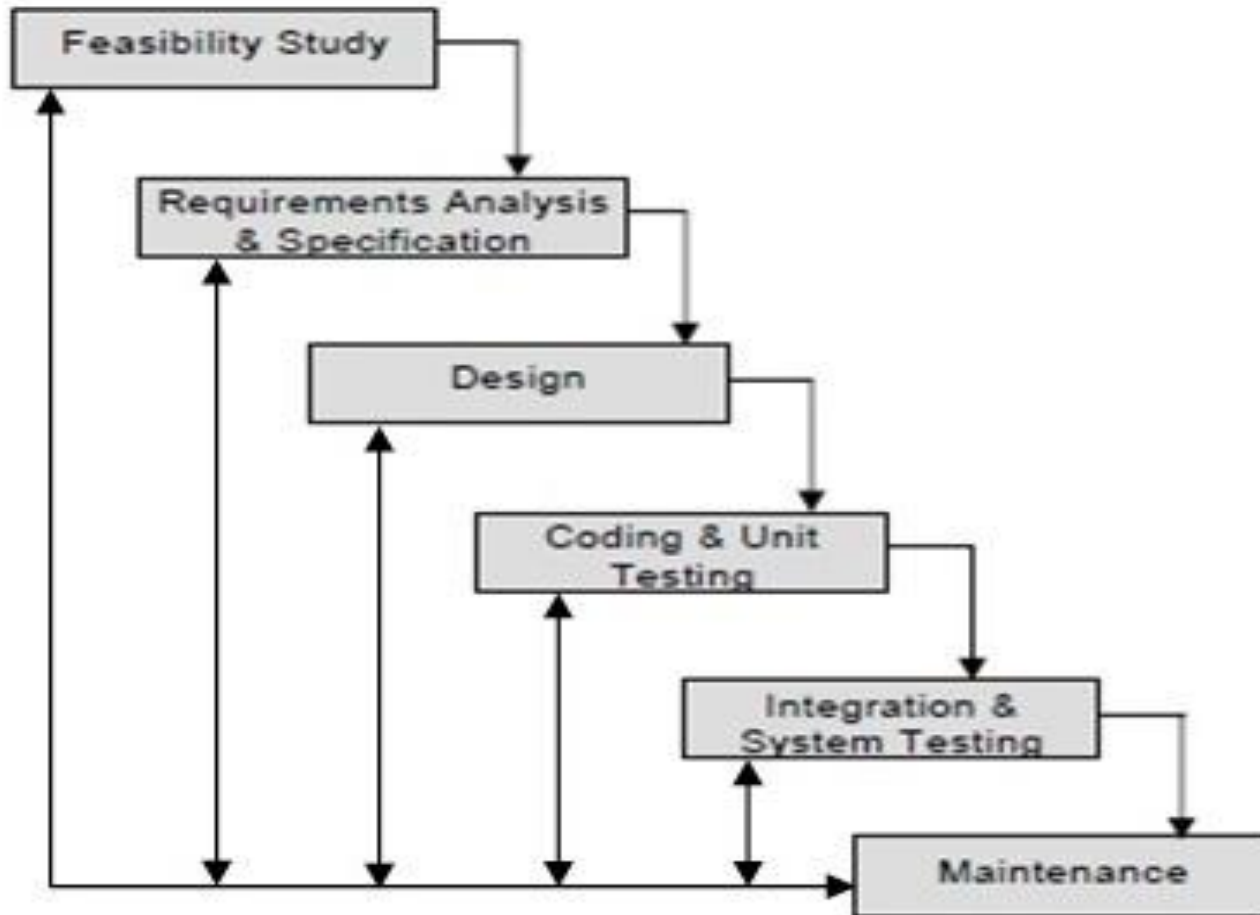
◆ Application:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- When the project is short.

1.6.2 Iterative Waterfall model

- Classical waterfall model is **idealistic**: it assumes that no defect is introduced during any development activity.
- But in practice: **defects do get introduced in almost every phase** of the life cycle. Even defects may get at much later stage of the life cycle.
- So, solution of this problem is → *iterative waterfall model*.
- Iterative waterfall model is by far the **most widely used model**. Almost every other model is derived from the waterfall model.
- The principle of **detecting errors as close to its point of introduction** as possible – is known as ***phase containment of errors***.
- Phase containment of errors can be achieved by reviewing after every milestone.

1.6.2 Iterative Waterfall model



Iterative waterfall model

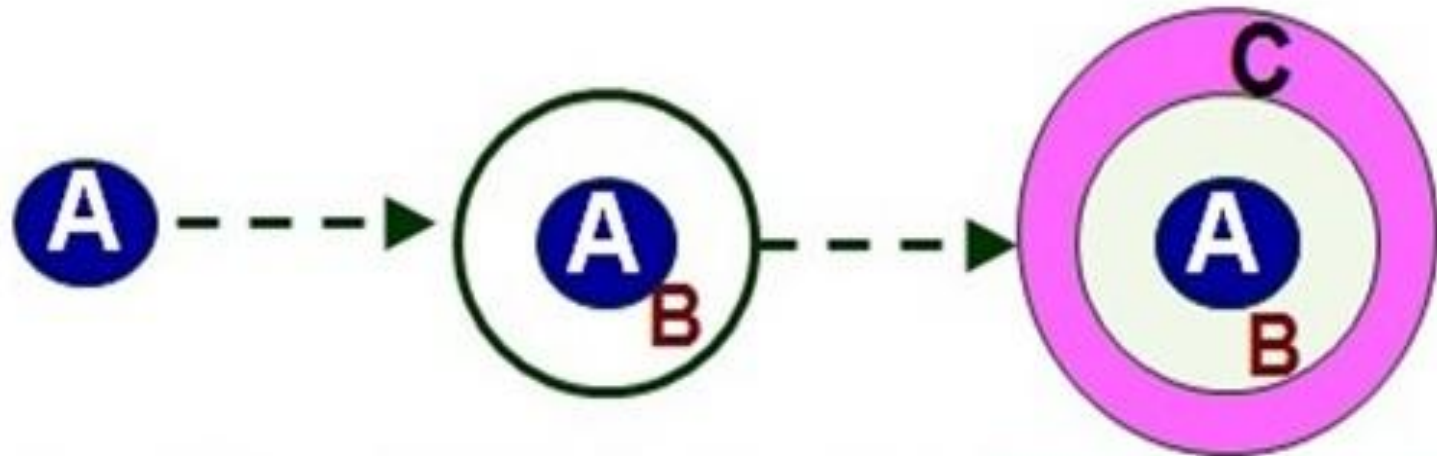
Continued...

- **Feedback paths** are added in classical waterfall model as shown in the figure.
- **Classical waterfall model with this feedback path** is known as a iterative waterfall model.
- In this model **Correction of errors is done** during the phase in **which they occur**.
- So in this model it is very **easy to handle the errors** compare to classical model..

1.6.3 Incremental model

- It is also referred as the *successive version of waterfall model* using incremental approach and *evolutionary model*.
- In this model, the system is **broken down into several modules** which can be incrementally implemented and delivered.
- **First develop the core product** of the system. The core product is used by customers to evaluate the system.
- The initial product skeleton is refined into increasing levels of capability: **by adding new functionalities in successive versions.**

1.6.3 Incremental model



Core module
of the system

Refined further and
add new functionality

Each version is
developed using Iterative
waterfall model

Incremental model

1.6.3 Incremental model

↳ Advantages:

- Each successive version more useful work than previous versions.
- Reducing chance of errors.
- **More flexible** and **less costly** to **change** the scope and **requirement**.
- User gets a chance to experiment with partially developed software.
- This model helps finding exact user requirements
- Feedback is useful for **better final** product .

↳ Disadvantages:

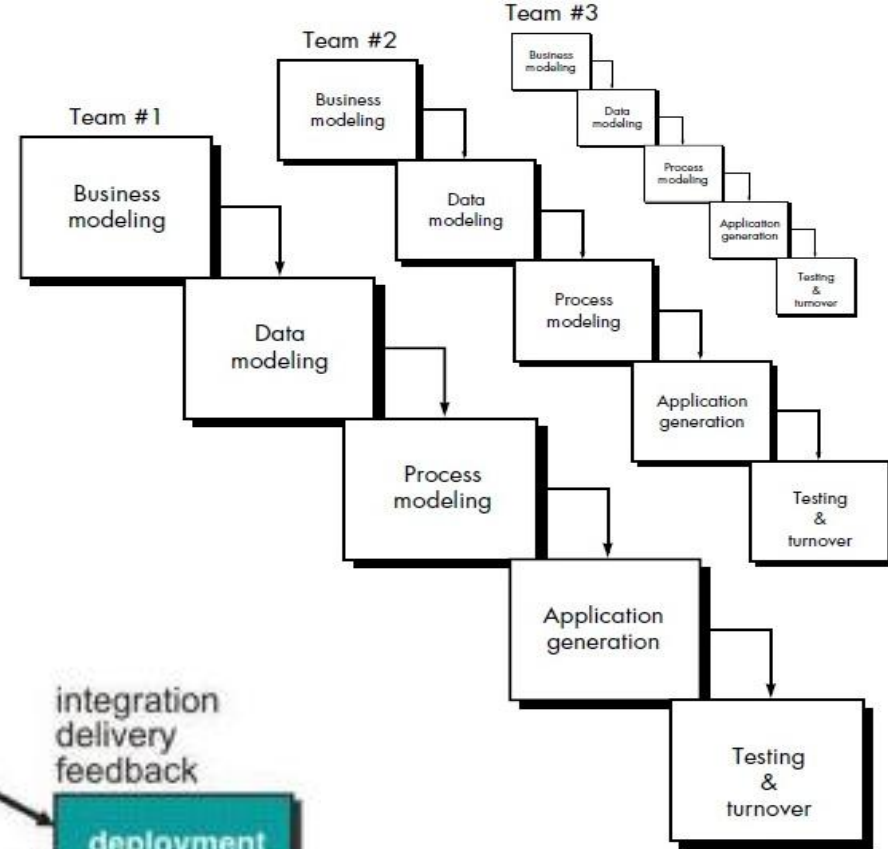
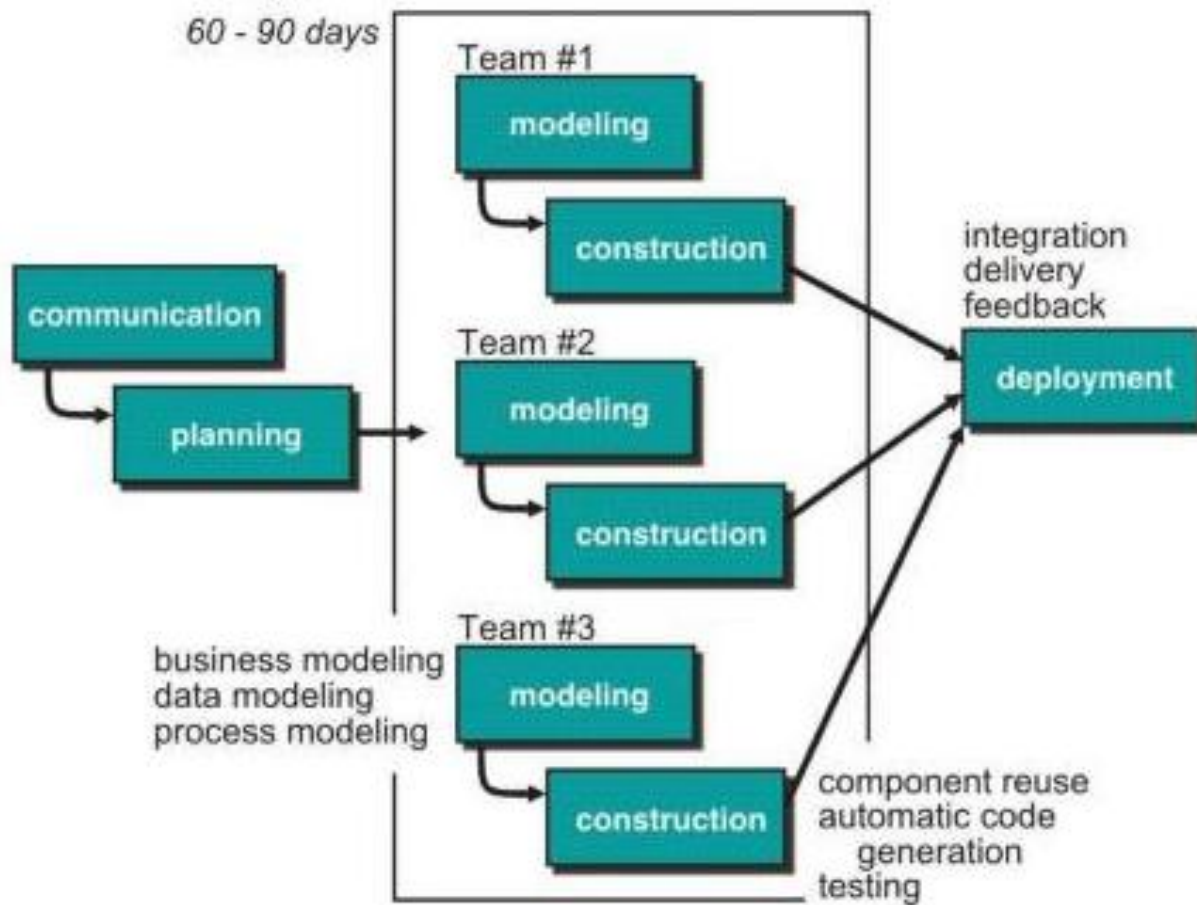
- Sometimes **difficult to subdivide** the problems into modules.
- Model can be used for very large problems.
- It **needs good planning** and design

↳ Application

- when large problems and **user requirement are not well specified**
- Needs **core functionality very fast**

1.6.4 RAD model

- It is proposed by IBM in 1980.
- It emphasizes an extremely short development cycle.
- It emphasize on *reuse*.
- In it rapid development is achieved by using component-based construction.
- If requirements are **well understood and project scope is constrained**, the RAD process enables a development team to create a “fully functional system” within **very short time periods** (e.g., **60 to 90 days**).
- **User involvement is essential** from requirement analysis to delivery.
- For this model, **requirements must be cleared** and well understood initially.
- Many development teams are **working parallel** to complete the task.



1.6.4 RAD model

RAD Model Phases

Activities performed in RAD Modeling

Business Modeling

- On basis of the flow of information and distribution between various business channels, the product is designed

Data Modeling

- The information collected from business modeling is refined into a set of **data objects** that are significant for the business

Process Modeling

- The data object that is declared in the data modeling phase is transformed to achieve the **information flow necessary to implement a business function**

Application Generation

- Automated tools are used for the construction of the software, to convert process and data models into **prototypes**

Testing and Turnover

- As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.

1.6.4 RAD model

↪ When to use RAD Methodology?

- When a system needs to be produced in a short span of time (2-3 months)
- When the requirements are known
- When the user will be involved all through the life cycle
- When technical risk is less
- When there is a necessity to create a system that can be modularized in 2-3 months of time
- When a budget is high enough to afford designers for modeling along with the cost of automated tools for code generation

↪ Advantages:

- Application can be developed in a quick time.
- This model highly makes use of reusable components.
- Reduce time for developing and testing.
- Customer satisfaction is improved due to full involvement.

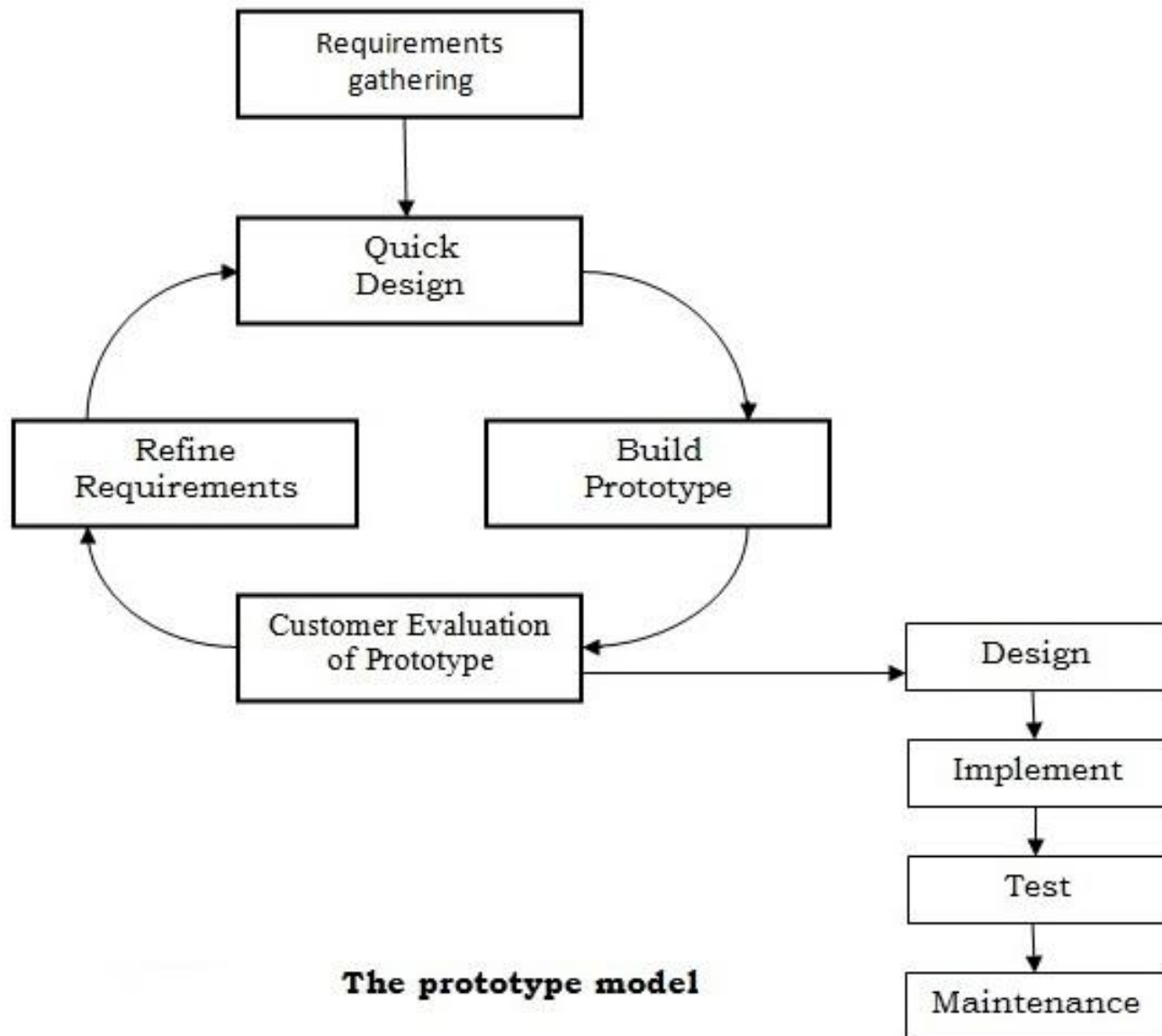
↪ Disadvantages:

- Requirements must be cleared and well understood for this model.
- It is not well suited where technical risk is high.
- In it, highly skilled and expert developers are needed.

1.6.5 Prototype model

- Prototype is a working physical system or sub system. Prototype is nothing but **a toy implementation** of a system.
- In this model, before starting actual development, a working prototype of the system should first be built.
- A prototype is actually a **partial developed** product.
- Compared to the actual software, a prototype usually have
 - limited functional capabilities
 - low reliability
 - inefficient performance
- It is built using several shortcuts.
- It is very useful in developing **GUI part** of system.
- It turns out to be a very crude version of the actual system.

1.6.5 Prototype model



1.6.5 Prototype model

- In working of the prototype model, product development starts with initial requirements gathering phase.
- Then, quick design is carried out and prototype is built.
- The developed prototype is then submitted to the customer for his evaluation.
- Based on customer feedback, the requirements are refined and prototype is modified.
- This cycle of obtaining customer feedback and modifying the prototype continues till the customers approve the prototype.
- The actual system is developed using the different phases of iterative waterfall model.
- There are two types of prototype model.
 - Rapid Prototype throw away model.
 - Evolutionary prototype model.

1.6.5 Prototype model

↳ **Advantages:**

- Customers can get a chance to have a look of the product.
- New requirements can be accommodate easily.
- Missing functionalities identified quickly.
- It provides better flexibility in design and development.
- More chance of user satisfaction.

↳ **Disadvantages:**

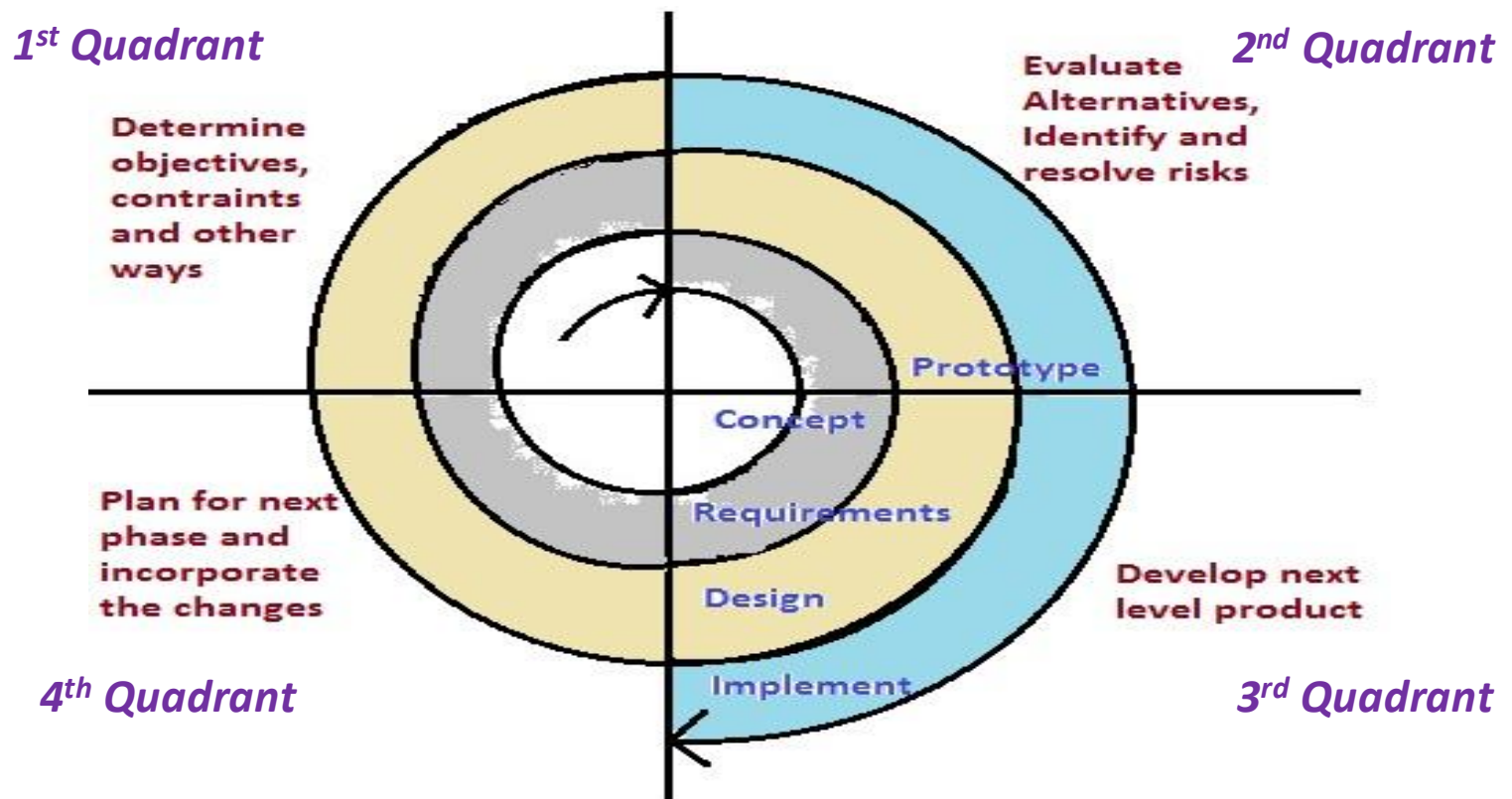
- **wasting of time** is there because core product is thrown away.
- The construction **cost** is very high.
- This model requires **extensive participation and involvement** of the customers that is not possible every time.
- If end user is not satisfied with the initial prototype, he may **lose interest** in the final product.

↳ **Application:**

- This model used when desired system **needs** to have a **lot of interactions with end users**.
- Generally used in **GUI based application**.

1.6.6 Spiral model

- This model is proposed by Boehm in 1986.
- In application development, spiral model uses fourth generation (4GL) languages and developments tools.
- The diagrammatic representation of this model appears like a spiral with many loops.



1.6.6 Spiral model

- Each loop of the spiral represents a phase of the software process:
 - the innermost loop might be concerned with system feasibility,
 - the next loop with system requirements definition,
 - the next one with system design, and so on.
- Phases:
 - 1. Planning
 - 2. Risk Analysis
 - 3. Engineering design
 - 4. Evaluation
 - 5. Planning
- This model is **more flexible** compared with other models.
- Each loop/phase in the spiral is split into **four sectors (quadrants)**
- 💧 *1st Quadrant: Determine objectives*
- Identifying the **objectives, their relationships** and find the **possible alternative solutions**.

1.6.6 Spiral model

💧 *2nd Quadrant: Identify and resolve risks*

- **Detailed analysis** is carried out of each identified risk.
- **Alternate solutions** are evaluated and risks are reduced at this quadrant. Prototype is built for best possible solution.

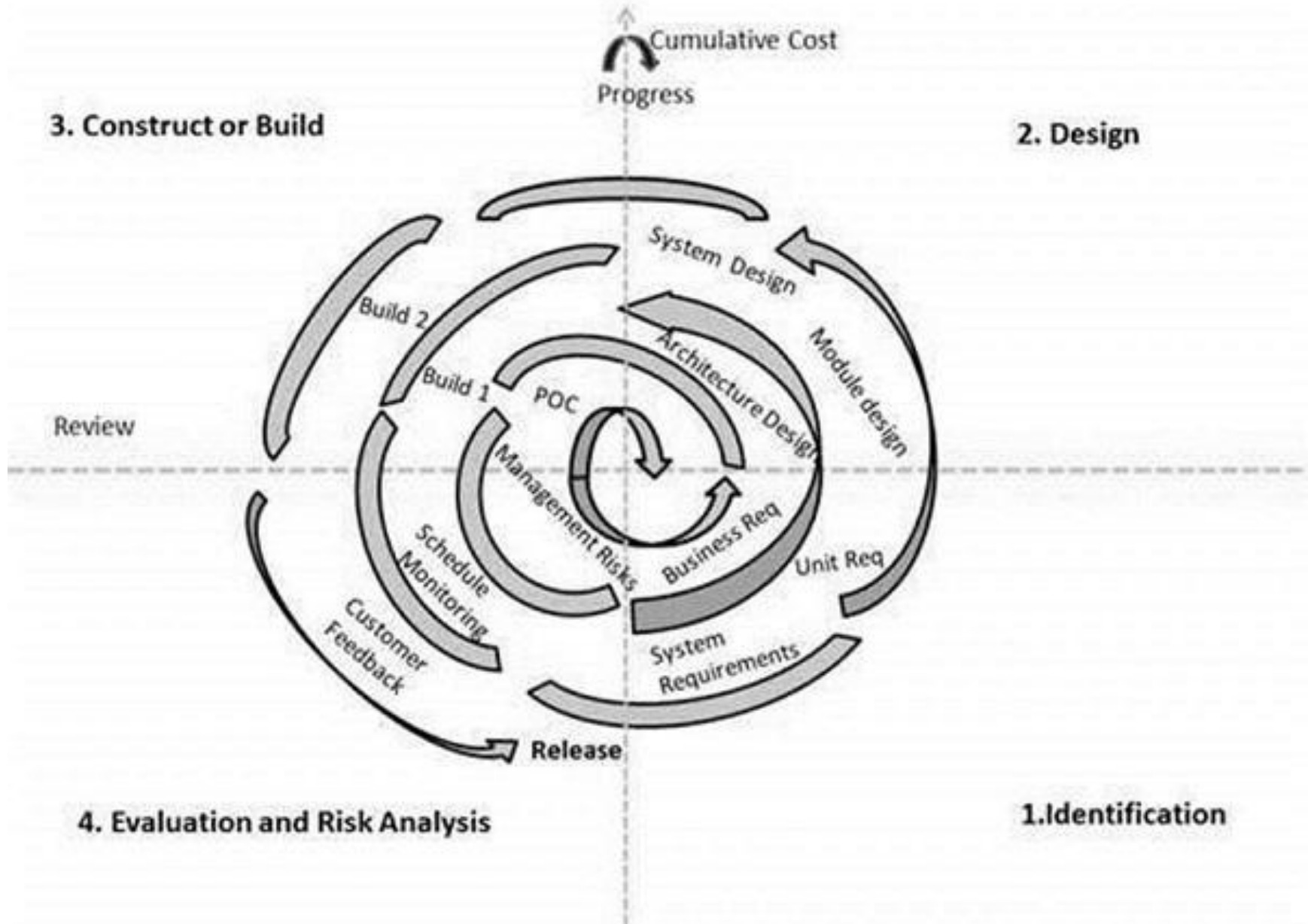
💧 *3rd Quadrant: Develop next level of product*

- **Develop and validate the next level of the product.** The identified features are developed and verified through testing
- **Activities** like design, code development, code inspection, testing and packaging are performed.

💧 *4th Quadrant: Customer evaluation (Review and Planning)*

- In this part, **review the results achieved** so far.
- **Plan the next iteration** around the spiral. Different plans like **development plan, test plan, installation plan** are performed.

1.6.6 Spiral model



1.6.6 Spiral model

- With each iteration around the spiral: progressively more complete version of the software gets built.
- In spiral model → at any point, **Radius** represents: cost and **Angular dimension** represents: progress of the current phase.
- At the end, all risks are resolved and s/w is ready to use.

Advantages:

- It is more flexible, as we can easily deal with changes.
- Due to user involvement, user satisfaction is improved.
- It provides cohesion between different stages.
- Risks are analyzed and resolved so final product will be more reliable.
- New idea and additional functionalities can be easily added at later stage.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

1.6.6 Spiral model

↳ *Disadvantages:*

- It is **more complex** to understand.
- It is applicable for **large problem** only.
- It can be **more costly** to use.
- More **number of documents** are needed as more number of spirals.
- Risk analysis phase requires much **higher expertise**.

1.6.6 Spiral model

↳ *Application:*

- When there is a **budget constraint and risk evaluation** is important.
- For medium to **high-risk projects**.
- Long-term project commitment because of **potential changes** to economic priorities as the **requirements change with time**.
- **Customer is not sure** of their requirements which is usually the case.
- **Requirements are complex** and need evaluation to get clarity.
- New product line which should be released in phases to get **enough customer feedback**.
- Significant **changes are expected** in the product during the development cycle.

1.6.6 Spiral model

☛ Spiral model can be viewed as a Meta model.

- Spiral model subsumes almost all the life cycle models.
- Single loop of spiral represents ***Waterfall Model***.
- Spiral model uses a ***prototyping approach*** by first building the prototype before developing actual product.
- The iterations along the spiral model can be considered as supporting the ***Evolutionary Model***.
- The spiral model retains the step-wise approach of the ***waterfall model***.




1.7 Agile Development Model

❖ Agile methodology

- A simple meaning of agile is → ready to move in quick and easy way.
- Agile methodology is a “step by step” dynamic focused on short-term visibility but never losing the long-term product goal.
- Divide the project into small phases and make it easily manageable.
- It adopt constant changes via repetitive or iterative approach.
- Concept of ‘Frame’ and working of iteration in agile development.
- At the end of each iteration, a working product with a new feature has to be delivered.
- Emphasizing incremental delivery, team collaboration, continual planning, and continual learning, instead of trying to deliver it all at once near the end.


1.7 Agile Development Model

PRINCIPLES OF AGILE





Satisfy the Customer

Welcome Changing Requirements




Deliver Working Software Frequently

Frequent Interaction with Stakeholders




Motivated Individuals

Face-to-Face Communication





Measure by working software

Maintain constant pace




Sustain technical excellence and good design

Keep it simple



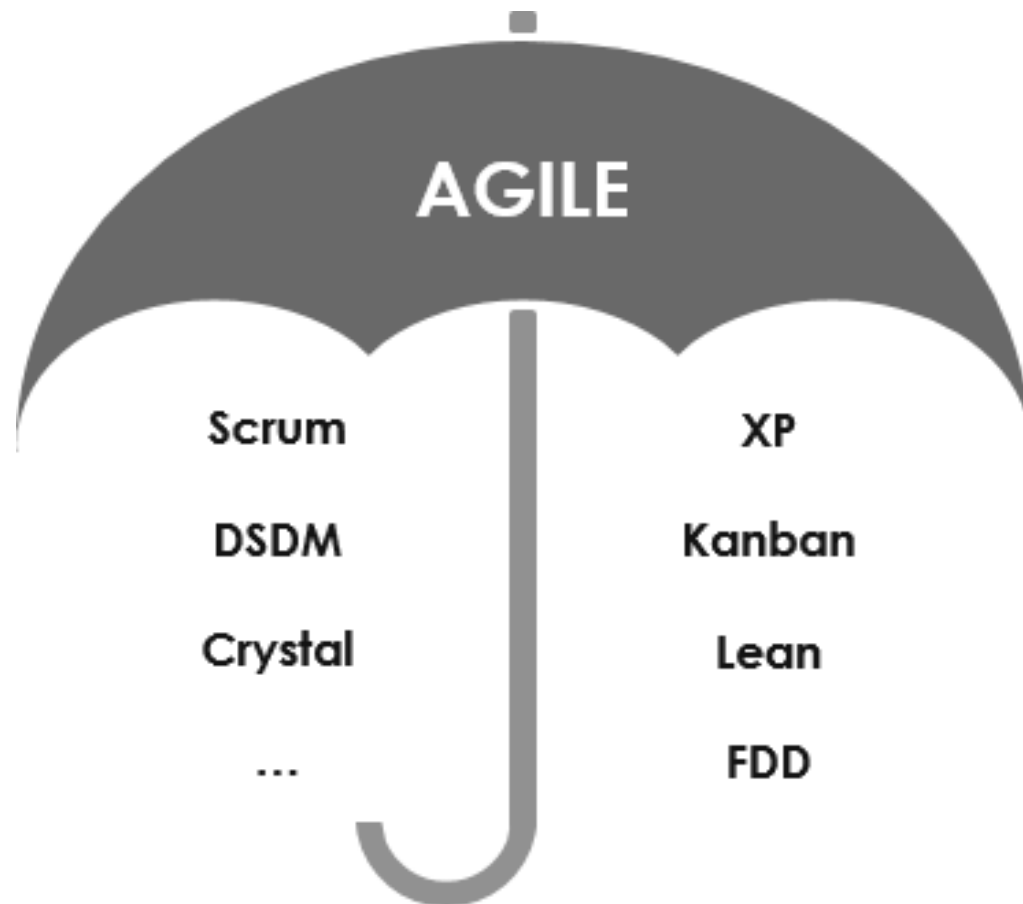
Empower self-organizing teams

Reflect and Adjust continuously



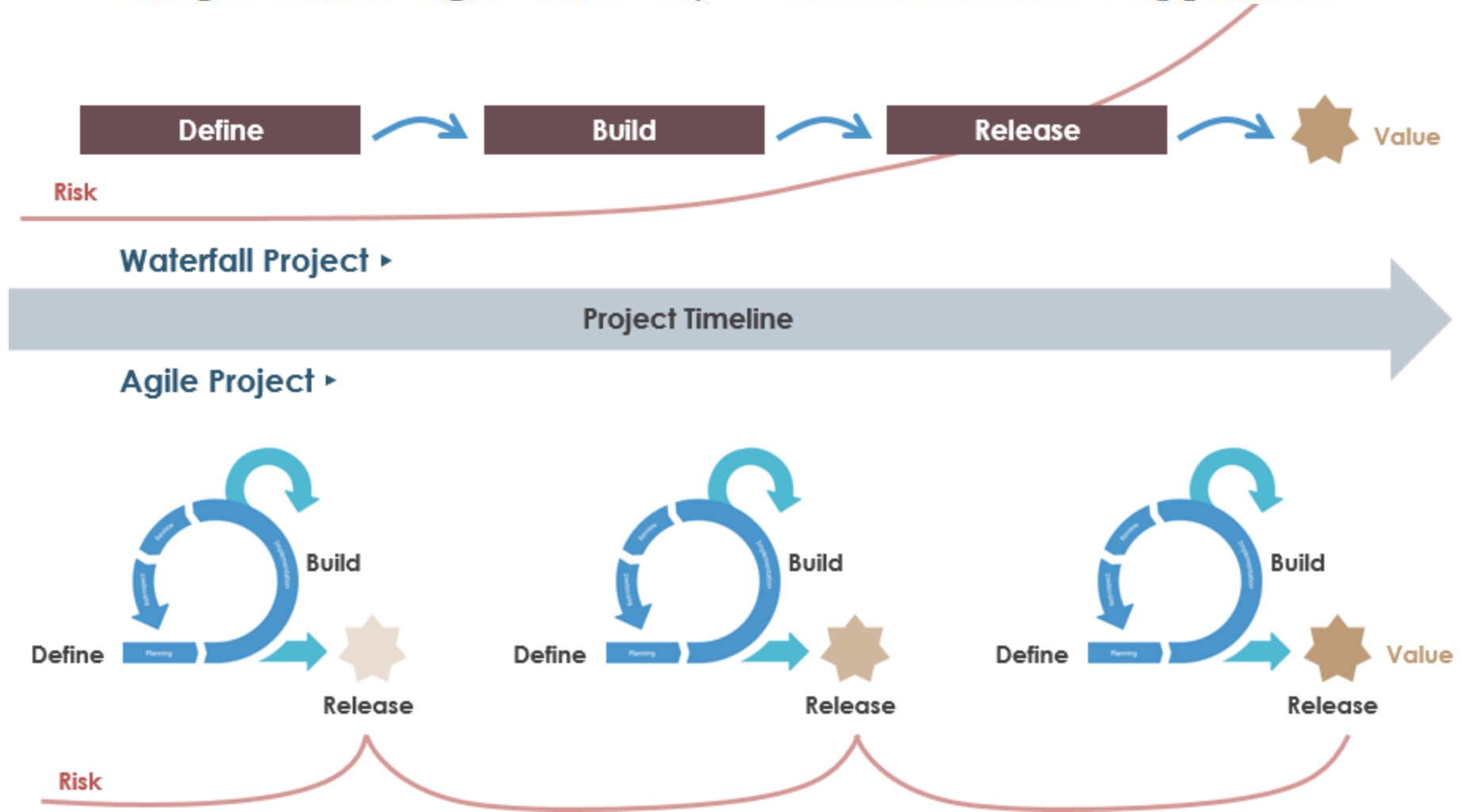
1.7 Agile Development Model

- Most popular and common agile methodology examples (or frameworks) are:



1.7 Agile Development Model

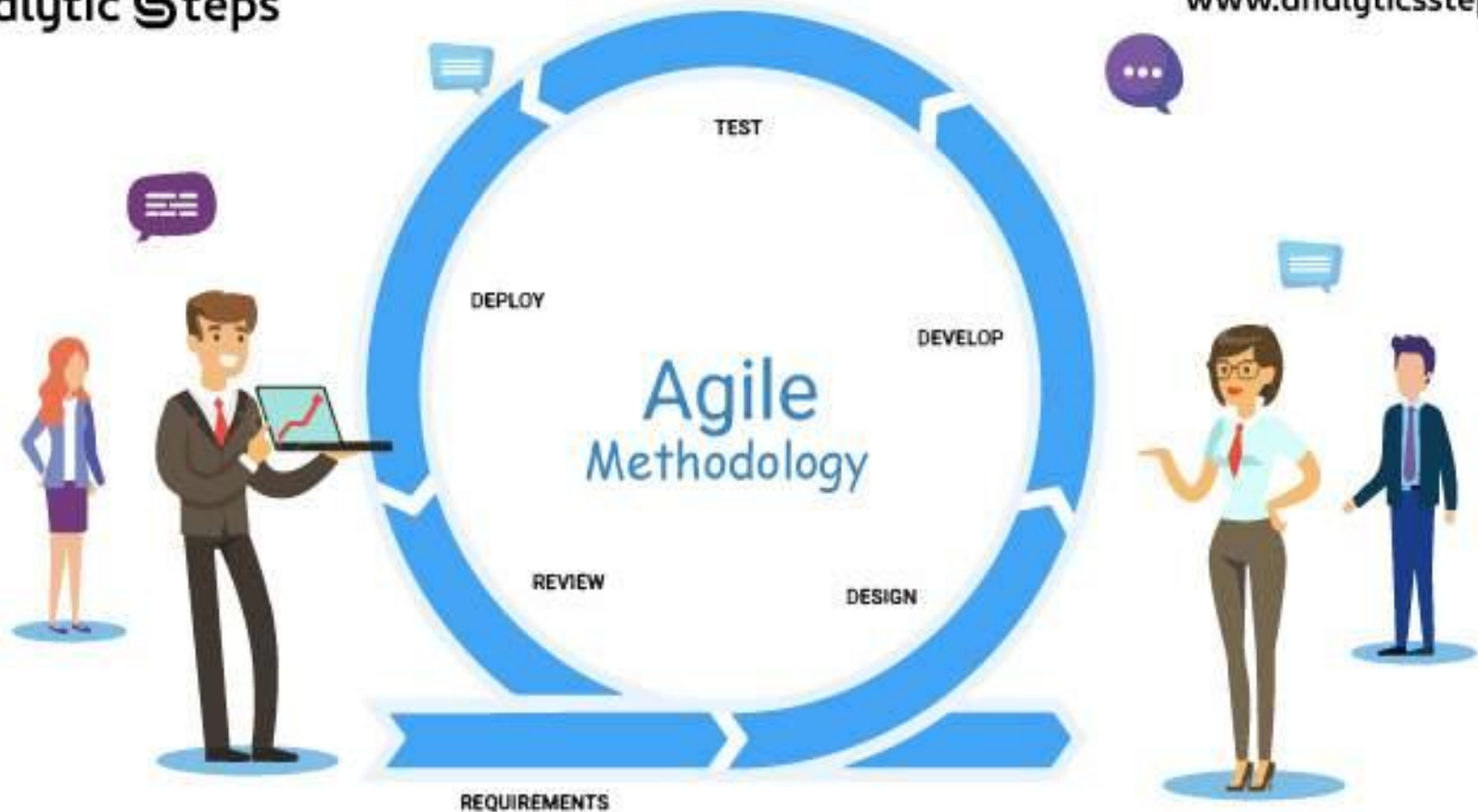
❖ Comparison of Agile model v/s Traditional model approach



1.7 Agile Development Model

 analytic Steps

www.analyticsteps.com





Scrum model

- It is a **light weight agile process framework**.
- Scrum is **adaptable, fast, flexible and effective agile** framework.
- It focused primarily on ways to manage tasks in **team-based development** conditions.
- This model based on an **iterative and incremental** processes.
- Scrum is characterized by cycles or stages of development, known as **'sprints'**
- Every day starts with a **small 15-minute meeting, called 'daily scrum'**, which takes the role of synchronizing activities and finding the best way to plan out the working day.

Works on Theme ,Epic and Story model

Scrum model

Example of movie theater:

Theme:

Fill empty seats in theaters

Epic:

Use a mobile app to drive last-minute ticket sales

Stories:

- Create and assign promotional codes for last-minute purchases
- Add text-message capability to the mobile app, to send last-minute promos and coupons
- Develop creative for promo emails and SMS texts

SCRUM

The Agile: Scrum Framework at a glance

Inputs from Executives,
Team, Stakeholders,
Customers, Users



Product Owner



The Team

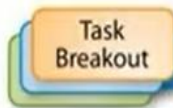


Product Backlog

Team selects starting at top as much as it can commit to deliver by end of Sprint

Sprint Planning Meeting

Estimating Stories (Story points)



Sprint Backlog



1-4 Week Sprint

Sprint end date and team deliverable do not change

Scrum Master

Burndown/up Charts



Every 24 Hours



Daily Scrum Meeting

Duration: 15 Mins

- Status 24 hours last/next
- Blockers if any



Sprint Review



Finished Work

Final Demo to PO/QA/DEV

- 1) What went well?
- 2) What went wrong?
- 3) Improvement Areas



Sprint Retrospective

Scrum model

Advantages of Scrum model

- It is adaptable and flexible.
- Focus on product quality.
- Improve team motivations.
- More chance of customer satisfaction.

Disadvantages of Scrum model

- Doesn't work well when more numbers of team members.
- Development team should be well experience.
- Daily meetings sometimes frustrate team members.
- This model is not appropriate for large and complex projects

Scrum model

→ Applications of Scrum model

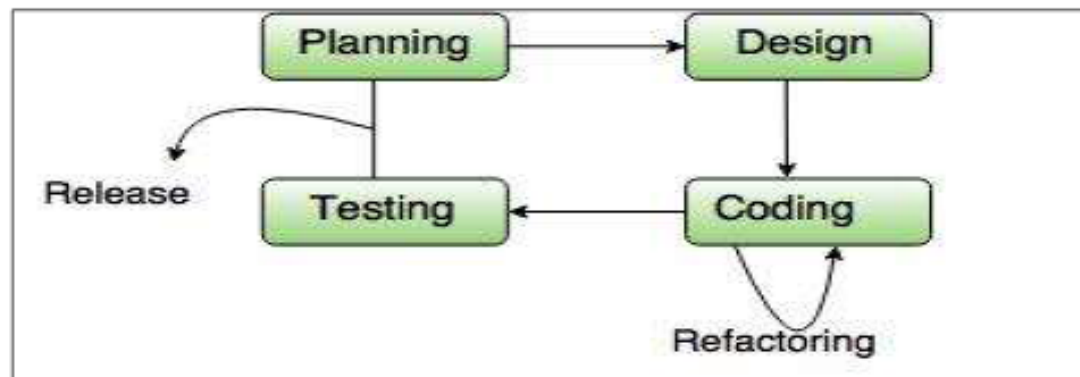
- Scrum model is a versatile framework that can be applied to a wide range of projects and industries like **healthcare, education, marketing, research etc.**
- Used when system have rapidly new changes are needed to be implementation required.
- Used when there is a **requirement of developing new features in quick time.**

eXtreme Programming (XP) model

- Developed by Kent Beck.
- Used when customers are constantly changing demands or requirements, or not sure about system's performance.
- XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- It advocates “frequent releases” of the product in short development cycles.
- Keep customer and quality in target.
- Team work is very important in this model.
- The sprint is released in very shorter life cycle (almost in 15-20 days).

eXtreme Programming (XP) model

- **XP is achieved with:**
 - ✓ Emphasis on continuous feedback from the customer
 - ✓ Short iterations
 - ✓ Design and redesign
 - ✓ Coding and testing frequently
 - ✓ Eliminating defects early, thus reducing costs
 - ✓ Keeping the customer involved throughout the development
 - ✓ Delivering working product to the customer



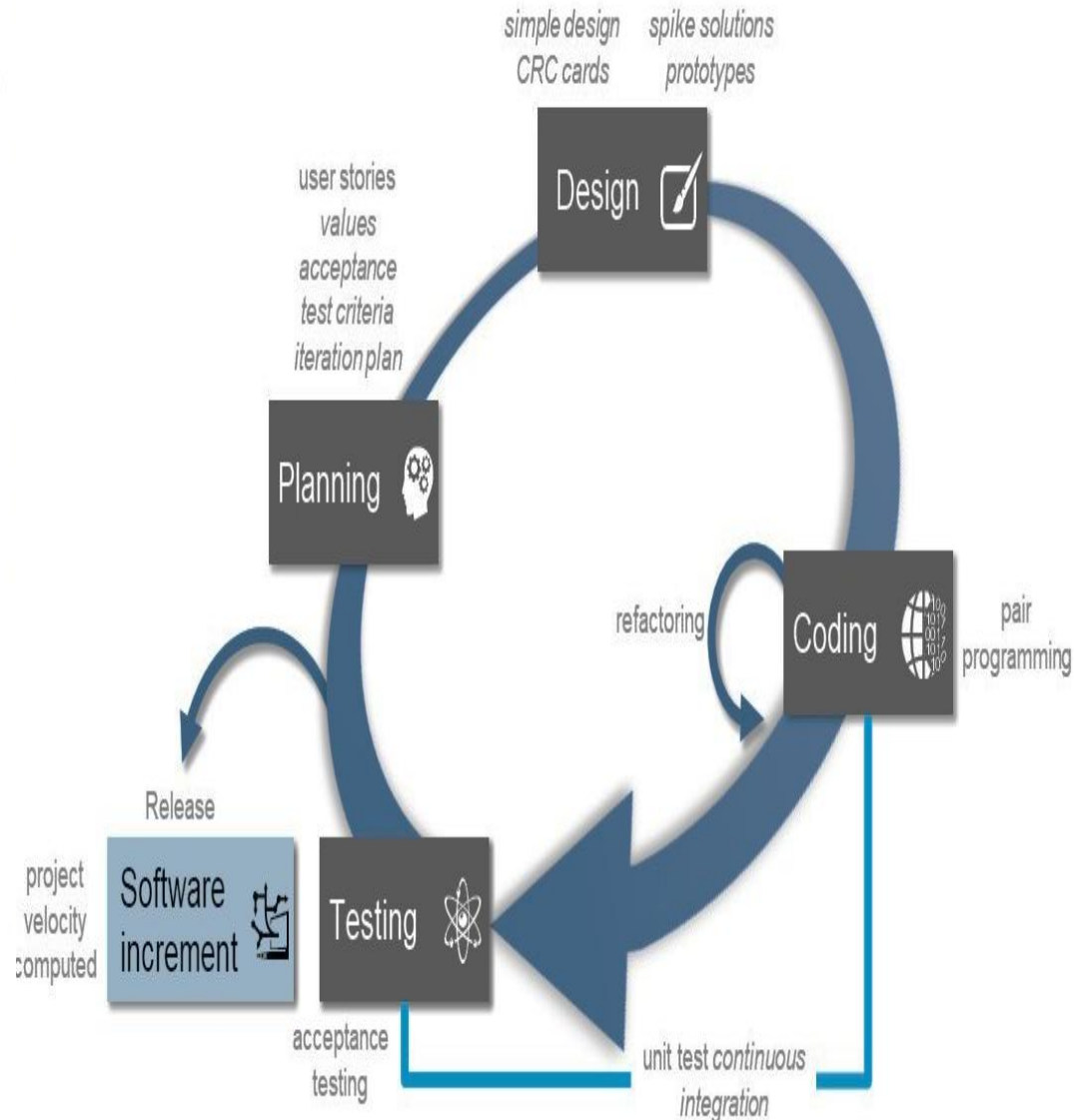
Explanation of each phase in fig.

• EXTREME PROGRAMMING(XP)

- XP is a lightweight , efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- Small to medium sized teams that work under vague and rapidly changing requirements.
- The five values of XP are communication, simplicity, feedback, courage, and respect
- Follows Object Oriented approach.

Spike solution: A **spike solution** is a very simple program to explore potential **solutions**. Difficult designs should be modeled using prototype.

Refactoring: Improves the internal structure of the code but external behavior not affected.



- Class-Responsibility Collaborator(CRC).

Class: <i>Librarian</i>	
Responsibilities	Collaborators
<i>check in book</i>	<i>Book</i>
<i>check out book</i>	<i>Book, Borrower</i>
<i>search for book</i>	<i>Book</i>
<i>knows all books</i>	
<i>search for borrower</i>	<i>Borrower</i>
<i>knows all borrowers</i>	

Class CardReader	
Responsibilities	Collaborators
Tell ATM when card is inserted	ATM
Read information from card	Card
Eject card	
Retain card	

ACCEPTANCE TESTING:-During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, System Testing is performed. Once System Testing is complete, Acceptance Testing is performed so as to confirm that the ballpoint pen is ready to be made available to the end-users.

eXtreme Programming (XP) model

- The XP model suggests quick release.
- Small releases allow developers to frequently receive feedback, detect bugs early, and monitor how the product works in production.

↳ **Advantages of XP model**

- stable and well-performing systems with minimal debugging.
- Less number of documents needed.
- Results can be derived soon.

↳ **Disadvantages of XP model**

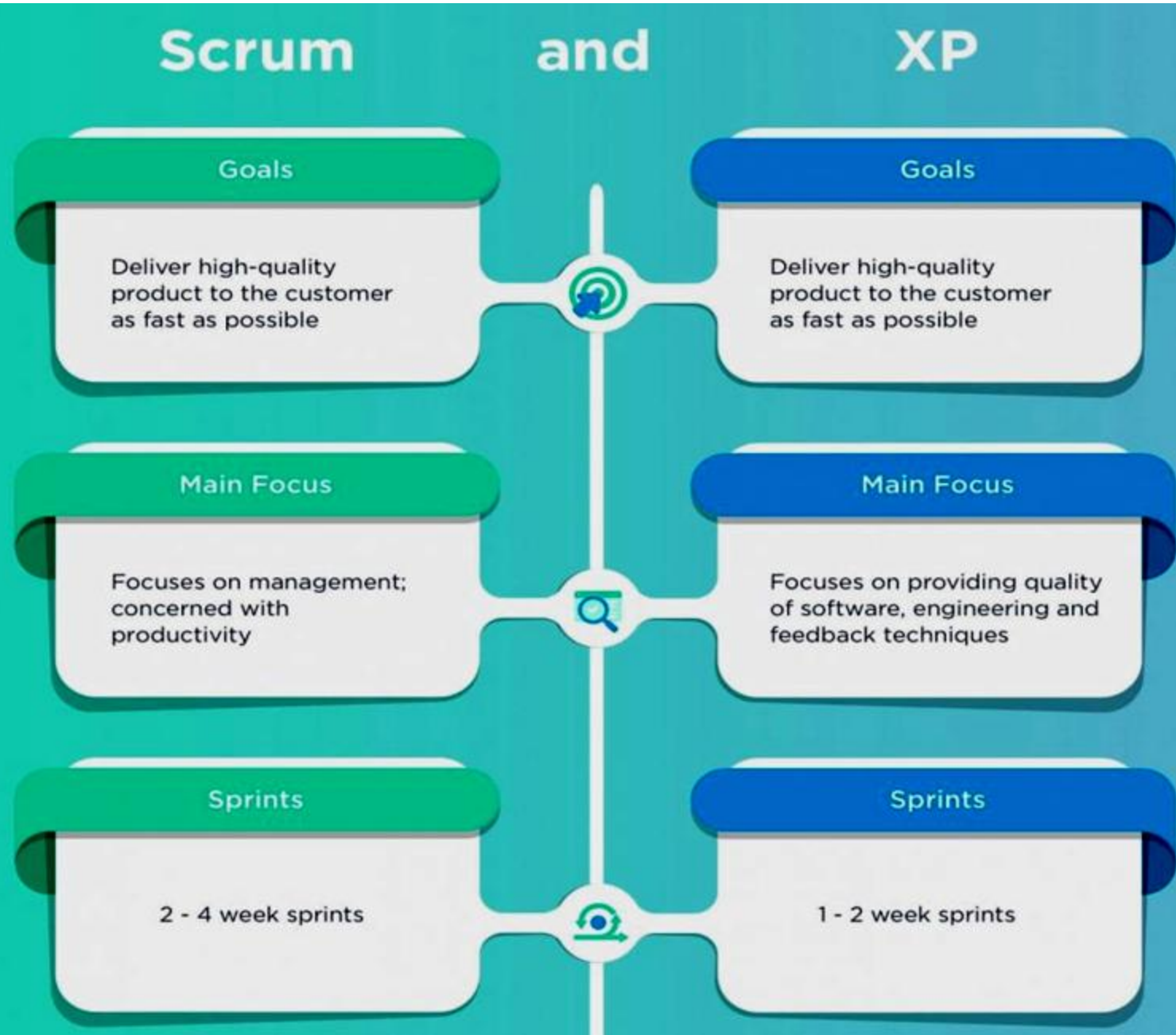
- Time consuming.
- Customer involvement is sometime tedious.
- High cost

eXtreme Programming (XP) model

↪ Applications of XP model

- Mainly used for smaller projects.
- Projects involving new technologies and research.

Scrum V/S Extreme Programming (XP)



Scrum V/S Extreme Programming (XP)

Accommodating Changes

No new changes allowed while sprint is in progress



Accommodating Changes

Flexibility; changes can be made by the customer during the sprint itself

Product Owner

Scrum Master communicates with the product owner during the development



Product Owner

Product owner communicates with the team of developers

Prioritizing Tasks

Product Owner determines the priority of the development tasks; developers determine the order of their actions themselves; Scrum team picks tasks in the sprint in order of their liking



Prioritizing Tasks

No flexibility

Values

Openness, focus, commitment



Values

Communication, simplicity, feedback

1.7 Agile Development Model

❖ Comparison of Agile model v/s Traditional model approach

Traditional model approach	Agile approach
- It is process oriented.	- It is people oriented.
- In it, project scope and requirements are less flexible.	- Project scope and requirements are more flexible.
- Clear initial requirements.	- Creative, innovative and unclear requirements.
- In this approach, planning is done at the beginning.	- In it, planning is done iteratively and incrementally.
- Focuses on delivering a complete product.	- Focuses on delivering working software or features in small increments.
- It is based on command and control project management.	- It is based on leadership and collaboration project management.
- In general – waterfall, spiral or prototype approach.	- In it – evolutionary approach.
- Does not involve customers continuously.	- Involvement of customers is continuous.
- This approach is better for bigger projects.	- This is better for smaller projects.

Comparison of different life cycle models

The Classical waterfall model

- Can be considered as a basic model.
- But, it is not used in practical development.

The Iterative waterfall model

- Most widely used model.
- But, suitable only for well-understood problems.

The Prototype model

- Suitable for projects in which user requirements and technical aspects are not well understood.
- Especially popular for user interface part of the project.

Comparison of different life cycle models

The incremental model

- It is suitable for large problems.
- used for object oriented development projects.
- But can be used only if the incremental delivery of the system is acceptable by the customer.

The Spiral model

- Used to develop the projects those have several kinds of risks.
- But, it is much complex than other models.

Agile model

- Used to develop the projects those have constant change in requirement.
- shorter life span
- Require quick product delivery.
- Problem statement is simple

Comparison

Factors	Waterfall	Evolutionary Prototyping	Spiral	Iterative and Incremental	Agile
Unclear User Requirement	Poor	Good	Excellent	Good	Excellent
Unfamiliar Technology	Poor	Excellent	Excellent	Good	Poor
Complex System	Good	Excellent	Excellent	Good	Poor
Reliable system	Good	Poor	Excellent	Good	Good
Short Time Schedule	Poor	Good	Poor	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Excellent	Excellent
Visibility of Stakeholders	Good	Excellent	Excellent	Good	Excellent
Skills limitation	Good	Poor	Poor	Good	Poor
Documentation	Excellent	Good	Good	Excellent	Poor
Component reusability	Excellent	Poor	Poor	Excellent	Poor

Thank YOU...