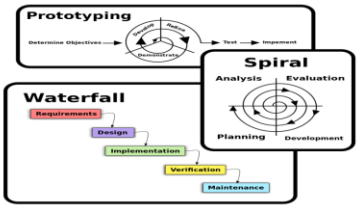
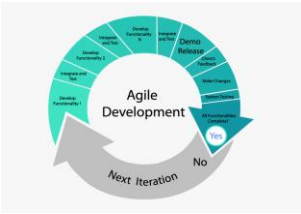
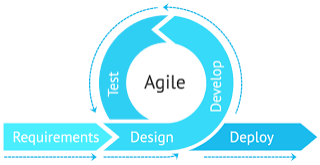


Software Engineering And Testing

Software Life Cycle Model





Outline

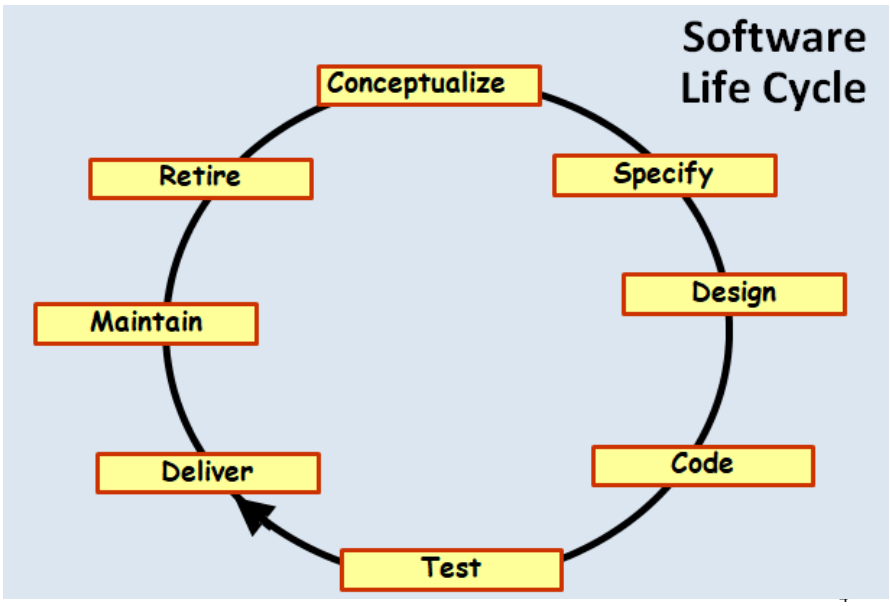
- Waterfall Model and its Extensions
- Classical Waterfall Model
- Iterative Waterfall Model
- V-Model
- Prototyping Model
- Incremental Development Model
- Evolutionary Model
- Rapid Application Development (RAD)
- Working of RAD
- Applicability of RAD Model
- Comparison of RAD with Other Models.
- Spiral Model
- Phases of the Spiral Model

Outline

- A Comparison of Different Life Cycle Models
- Selecting an Appropriate Life cycle Model for a Project
- Agile Development Models
- Essential Idea behind Agile Models
- Agile versus Other Models
- Extreme Programming Model
- Scrum Model.

3

Life Cycle Models



Life Cycle Models

- The life cycle of a software represents the series of identifiable stages through which it evolves during its life time.
- *Software life cycle has been defined to imply the different* stages (or phases) over which a software evolves from the initial customer request for it, to a fully developed software, and finally to a stage where it is no longer useful to any user, and is discarded.
- This stage where the customer feels a need for the software and forms rough ideas about the required features is known as the ***inception stage***.
- Once installed and made available for use, the users start to use the software. This signals the start of the operation (also called *maintenance) phase.*

5

Life Cycle Models

- *T*he maintenance phase usually involves continually making changes to the software to accommodate the bug-fix and change requests from the user.
- A *software development life cycle (SDLC) model (also called software life cycle model and software development process model) describes the different activities that need to be carried* out for the software to evolve in its life cycle.
- **Process *versus methodology :*** A software development process has a much broader scope as compared to a software development methodology. A process usually describes all the activities starting from the inception of a software to its maintenance and retirement stages, or at least a chunk of activities in the life cycle. It also recommends specific methodologies for carrying out each activity. A methodology, in contrast, describes the steps to carry out only a single or at best a few individual activities.

6

Life Cycle Models

- **Why use a development process?**
  - The primary advantage of using a development process is that it encourages development of software in a systematic and disciplined manner.
  - Software development organisations have realised that adherence to a suitable life cycle model helps to produce good quality software and also helps to minimise the chances of time and cost overruns.
  - Programming-in-the-small refers to development of a toy program by a single programmer. On the other hand, programming-in-the-large refers to development of professional software through team effort.

7

Life Cycle Models

- **Why use a development process?**
  - While development of software of the former type could succeed even when an individual programmer uses a build and fix style of development, use of a suitable SDLC is essential for a professional software development project involving team effort to succeed.
- **Why document a development process?**
  - A documented development process forms a common understanding of the activities to be carried out among the software developers and helps them to develop software in a systematic and disciplined manner. A documented development process model, besides preventing the misinterpretations that might occur when the development process is not adequately documented, also helps to identify inconsistencies, redundancies, and omissions in the development process.

8

Why Use Life Cycle Models?

- **A graphical and written description:**
  - Helps common understanding of activities among the software developers.
  - Helps to identify inconsistencies, redundancies, and omissions in the development process.
  - Helps in tailoring a process model for specific projects.
- **A software project will never succeed if:**
  - one engineer starts writing code,
  - another concentrates on writing the test document first,
  - yet another engineer first defines the file structure
  - another defines the I/O for his portion first.

9

Importance of Correct Model

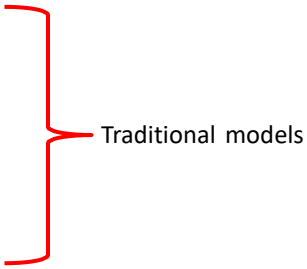
Importance of Correct Model

- **A graphical and written description:**
  - Helps common understanding of activities among the software developers.
  - Helps to identify inconsistencies, redundancies, and omissions in the development process.
  - Helps in tailoring a process model for specific projects.
- **A software project will never succeed if:**
  - one engineer starts writing code,
  - another concentrates on writing the test document first,
  - yet another engineer first defines the file structure
  - another defines the I/O for his portion first.

11

Life Cycle Models

- **Many life cycle models have been proposed.**
- **We confine our attention to only a few commonly used models.**
  - Waterfall
  - V model,
  - Evolutionary,
  - Prototyping
  - Spiral model,
  - Agile models



12

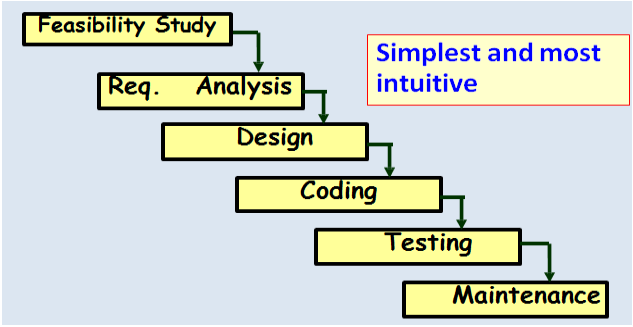
Software Life Cycle (software process)

- Series of identifiable stages that a software product undergoes during its life time:
  - Feasibility study
  - Requirements analysis and specification,
  - Design,
  - Coding,
  - Testing,
  - Maintenance.

13

Classical Waterfall Model

- **Classical waterfall model divides life cycle into following phases:**
  - Feasibility study
  - Requirements analysis and specification,
  - Design,
  - Coding and unit testing,
  - Integration and system testing,
  - Maintenance.



Phases of the waterfall model

- **Requirements**. A technical requirements document, also known as a functional specification, is created by analyzing potential needs, timeframes, and project parameters. The project is defined and planned at this stage of development, but no specific processes are mentioned.
- **Analysis.** To create product models and business logic to direct manufacturing, the system specifications are examined. At this point, the viability of the available financial and technological resources is also examined.
- **Design.** To describe the technical requirements for the design, such as the programming language, hardware, data sources, architecture, and services, a design specification document is made.

15

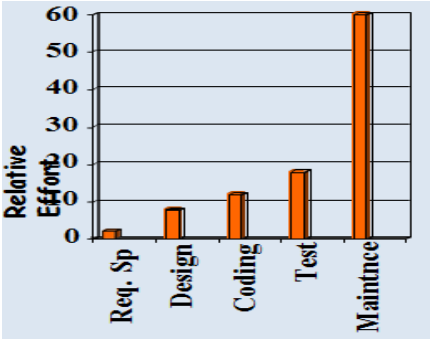
Phases of the waterfall model

- **Coding and implementation.** The models, logic, and requirement specifications specified in the earlier phases are used to develop the source code. The system is typically coded in smaller pieces or units before being assembled.
- **Testing.** At this point, problems that need to be fixed are found through quality assurance, unit, system, and beta testing. This may need repeating the debugging phase of the development process. If the system succeeds in development and testing, the waterfall process continues.
- **Operation and deployment.** The product or application is released into a live environment after being determined to be fully functional.
- **Maintenance.** To continuously improve, update, and improve the product and its functioning, corrective, adaptive, and corrective maintenance is carried out. Release of patch updates and fresh versions may fall under this category.

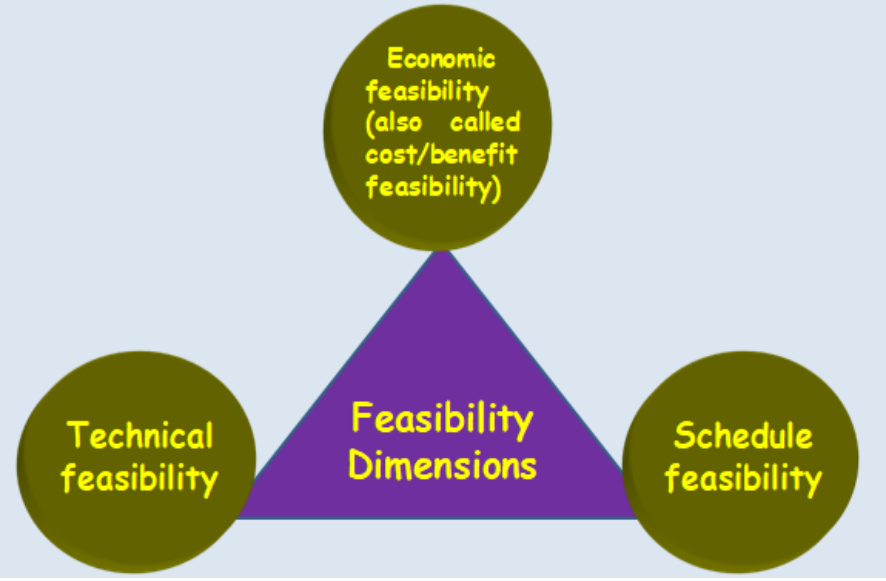
16

Relative Effort for Phases

- **Phases between feasibility study and testing**
  - Called development phases.
- **Among all life cycle phases**
  - Maintenance phase consumes maximum effort.
- **Among development phases,**
  - Testing phase consumes the maximum effort.



Feasibility Study



18

Feasibility Study

Feasibility Study

A Feasibility Study Report Answers the Following Questions

- Does the company's technical competence enough to overcome project challenges?
- Does the company have sufficient funds to conduct project works?
- What are the legal provisions related to the project?
- What kind of risks can exist in undertaking the project?
- Does the proposed scope meet the organization's objectives and stakeholder's requirements?
- Can the project be completed on the proposed time?

21

Feasibility Study

Legal

- Legal feasibility is an analysis performed to understand if the proposed plan conforms to the legal and ethical requirements. These requirements may involve zoning laws, data protection acts, or social media laws, etc.
- For example, your company is planning to open a branch in a new region. According to the studies you recognize that the country does not allow an individual foreigner owning a property. Therefore you select the rental option instead of buying.

22

Feasibility Study

Economic

- An economic feasibility study involves a cost benefits analysis to identify how well, or how poorly, a project will be concluded. You calculate the expected monetary value of each cost and benefit separately to decide if the project is economically feasible or not.
- For example, your company is planning to perform a housing project on the west coast of the city. In order to understand if the project is economically feasible, you will calculate the duration, cost, and income of the project. If the calculations demonstrate a short payback period, the board of directors will decide to undertake the project.

23

Feasibility Study

Technical

- Technical feasibility is a broad concept that can be applied to a software development project, pipeline construction project, or a military project. Each project requires different technical specifications and standards. Technical feasibility is the process of validating the technical resources and capabilities to convert the ideas into working systems.
- For example, your company is planning to improve the current network infrastructure. You analyzed the new system and concluded that the new system can use the organization's existing network infrastructure. This shows that a new system is technically feasible.

24

Feasibility Study

Operational

- Operational feasibility is performed to understand well a proposed system solves the problems. From this aspect, operational feasibility studies are critical for system engineering and affect early design phases.
- For example, your company has undertaken a project to build a new theme park for a client. Then you performed a study to determine how the theme park will operate in a way that is conducive to its inhabitants, parking, dining, human flow, accessibility. This can be an example of operational feasibility.

25

Feasibility Study

Scheduling (Time)

- Completing a project on time is very important from an investor's perspective. Scheduling feasibility is a measure of how reasonable the project duration is. If the project takes longer than estimated, investors will have to bear extra costs.
- For example, an investor proposed a hotel construction project to your company. However, he requested that the project will be completed in one year. The project team conducted a feasibility study based on a list of requirements to complete the project on time.

26

Feasibility Study First Step

- **Main aim of feasibility study: determine whether developing the software is:**
  - Financially worthwhile
  - Technically feasible.
- **Roughly understand what customer wants:**
  - Data which would be input to the system,
  - Processing needed on these data,
  - Output data to be produced by the system,
  - Various constraints on the behavior of the system.
- Example : Taking Admission to college

27

Activities During Feasibility Study

- **Work out an overall understanding of the problem.**
- **Formulate different solution strategies.**
- **Examine alternate solution strategies in terms of:**
  - resources required,
  - cost of development, and
  - development time.
- **Perform a cost/benefit analysis:**
  - Determine which solution is the best.
  - May also find that none of the solutions is feasible due to:
    - high cost,
    - resource constraints,
    - technical reasons.

28

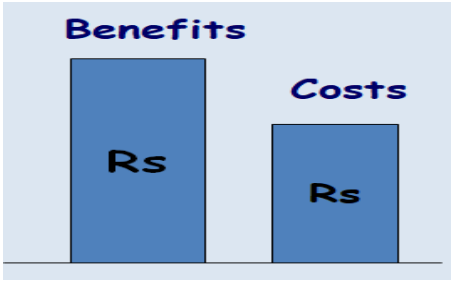
Cost benefit analysis (CBA)

- **Need to identify all costs — these could be:**
  - Development costs
  - Set-up
  - Operational costs
- **Identify the value of benefits**
- **Check benefits are greater than costs**

29

The business case

- **Benefits of delivered project must outweigh costs**
- **Costs include:**
  - Development
  - Operation
- **Benefits:**
  - Quantifiable
  - Non-quantifiable



- **Feasibility studies should help write a 'business case'**
- **Should provide a justification for starting the project**
- **Should show that the benefits of the project will exceed:**
  - **Various costs**
- **Needs to take account of business risks**

30

Design

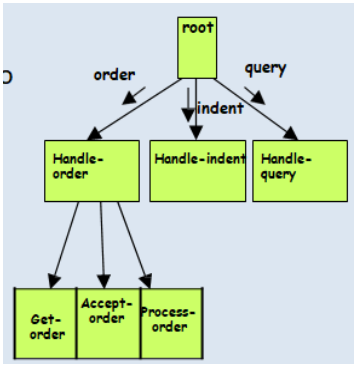
- **During design phase requirements specification is transformed into :**
  - A form suitable for implementation in some programming language.
- **Two commonly used design approaches:**
  - Traditional approach,
  - Object oriented approach

31

Design

- Traditional approach, Consists of two activities:

1. Structured analysis (typically carried out by using DFD)
2. Structured design
   - **High-level design:**
     - decompose the system into modules,
     - represent invocation relationships among modules
   - **Detailed design:**
     - different modules designed in greater detail:
     - data structures and algorithms for each module are designed.



Design

- Object- Oriented Design
- **First identify various objects (real world entities) occurring in the problem:**
  - Identify the relationships among the objects.
  - For example, the objects in a pay-roll software may be:
    - employees, Managers, pay-roll register, etc.
  - **Object structure:** Refined to obtain the detailed design.
  - **OOD has several advantages:**
    - Lower development effort,
    - Lower development time,
    - Better maintainability.

33

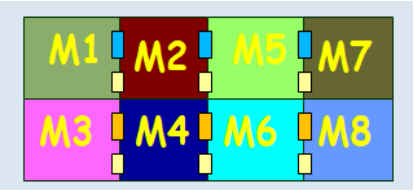
Coding and Unit Testing

- **During this phase:**
  - Each module of the design is coded,
  - Each module is unit tested
  - That is, tested independently as a stand alone unit, and debugged.
  - Each module is documented.

34

Integration and System Testing

- **Different modules are integrated in a planned manner:**
  - Modules are usually integrated through a number of steps.



  - **During each integration step,**
    - the partially integrated system is tested.
- **After all the modules have been successfully integrated and tested:** System testing is carried out.
- **Goal of system testing:**
  - Ensure that the developed system functions according to its requirements as specified in the SRS document.

33

System Testing

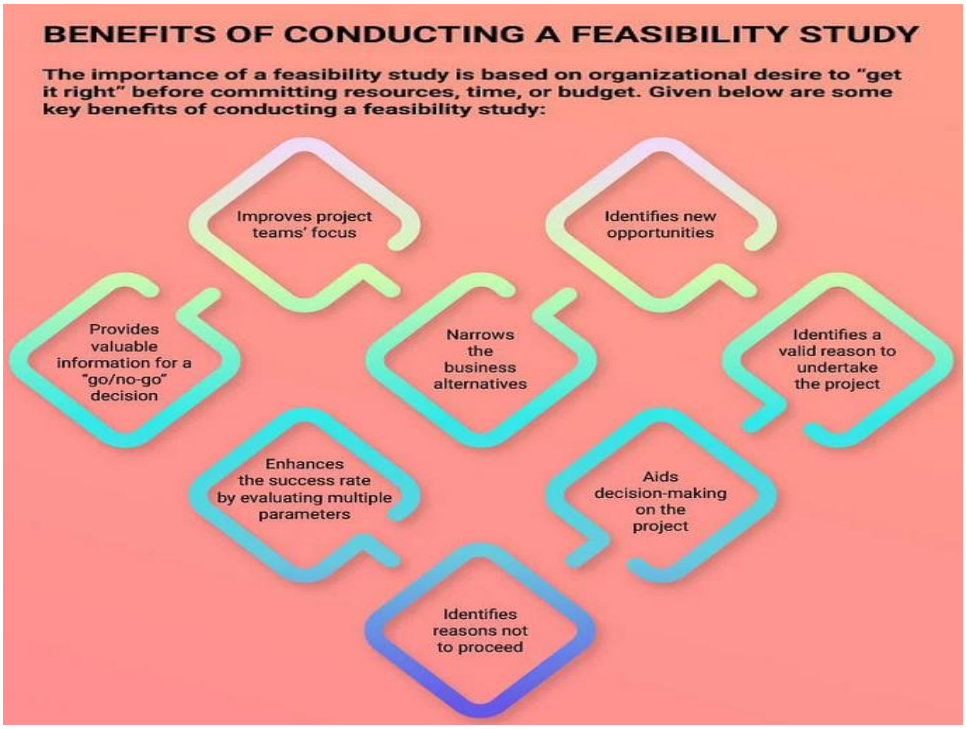
- **System testing usually consists of three different kinds of testing activities:**
  - □α-testing: a testing is the system testing performed by the development team.
  - □β-testing: This is the system testing performed by a friendly set of customers.
  - □Acceptance testing: After the software has been delivered, the customer performs system testing to determine whether to accept the delivered software or to reject it.

36

Maintenances

- **Maintenance of any software:**
  - Requires much more effort than the effort to develop the product itself.
  - Development effort to maintenance effort is typically 40:60.
- **Types of Maintenance?**
  - **Corrective maintenance:** Correct errors which were not discovered during the product development phases.
  - **Perfective maintenance:** Improve implementation of the system to enhance functionalities of the system.
  - **Adaptive maintenance:** Port software to a new environment, e.g. to a new computer or to a new operating system.

37



Waterfall Models

Classic life cycle or linear sequential model

- When **requirements** for a problems are **well understood** then this model is used in which **work flow** from feasibility study to deployment & maintenance is **linear**.

When to use ?

- **Requirements** are very well **known**, **clear** and **fixed**
- Product **definition** is **stable**
- **Technology** is **understood**
- There are **no ambiguous** (unclear) **requirements**
- Ample (**sufficient**) **resources** with required **expertise** are **available** freely
- The **project** is **short**

Advantage

- **Simple to implement** and manage

39

Waterfall Models

Classic life cycle or linear sequential model

- When **requirements** for a problems are **well understood** then this model is used in which **work flow** from feasibility study to deployment & maintenance is **linear**.

When to use ?

- **Requirements** are very well **known**, **clear** and **fixed**
- Product **definition** is **stable**
- **Technology** is **understood**
- There are **no ambiguous** (unclear) **requirements**
- Ample (**sufficient**) **resources** with required **expertise** are **available** freely
- The **project** is **short**

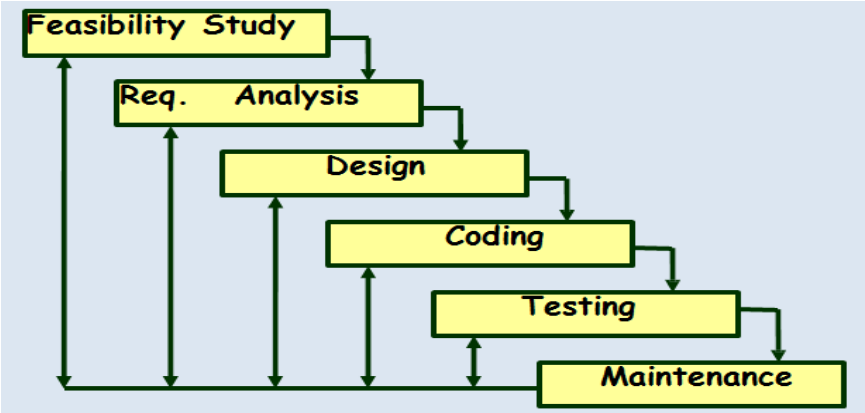
Advantage

- **Simple to implement** and manage

40

Iterative Waterfall Model

- **Classical waterfall model is idealistic:**
  - Assumes that no defect is introduced during any development activity.
  - In practice: Defects do get introduced in almost every phase of the life cycle.

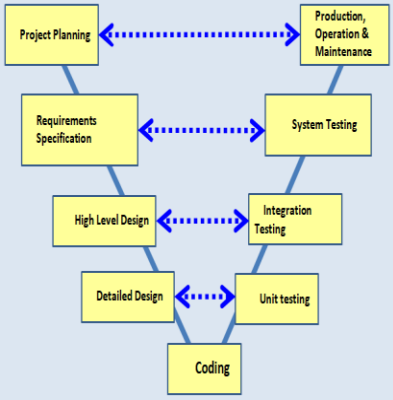


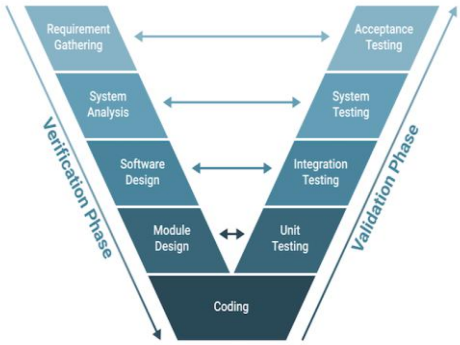
Phase Containment of Errors

- The principle of detecting errors as close to their points of commitment as possible is known as phase containment of errors.
- **Errors should be detected:** In the same phase in which they are introduced.
- **For example:** If a design problem is detected in the design phase itself,
  - The problem can be taken care of much more easily.
  - Than say if it is identified at the end of the integration and system testing phase.
- **A Reason: rework must be carried out not only to the design but also to code and test phases.**

V Model

- **It is a variant of the Waterfall**
  - emphasizes verification and validation
  - V&V activities are spread over the entire life cycle.
- **In every phase of development:**
  - Testing activities are planned in parallel with development.





When to use the V-Model Model?

- Due to the nature of the V-Model, it is hard to back to the previous phase once completed.
- Although, this is can be very rigid in some software projects which need some flexibility, while, this model can be essential or the most suitable model to other software projects' contexts and which focus on quality as an important aspect of the delivery.
- The usage of V-Model can fall under the projects which not focus on changing the requirements, for example:

1. Projects initiated from a request for proposals (RFPs), the customer has a very clear documented requirements
2. Military projects
3. Mission Critical projects, for example, in a Space shuttle
4. Embedded systems.
5. Projects with defined and clear requirements

V-Model Advantages

- Simple and easy to use
- Each phase has specific deliverables.
- Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
- Works well for where requirements are easily understood.
- V-Model Improves the quality and reliability of the software.
- It reduces the amount of re-work because of the early detection of defects and issues.
- It provides better management for project risks.
- Verification and validation of the product in the early stages of product development ensures better quality.
- The V-Model concept can be combined with other models, for example, the iterative and agile models.

45

V-Model Disadvantages

- Very inflexible, like the waterfall model.
- Adjusting scope is difficult and expensive.
- The software is developed during the implementation phase, so no early prototypes of the software are produced.
- The model doesn't provide a clear path for problems found during testing phases.
- Moreover, It is costly and required more time, in addition to a detailed plan.

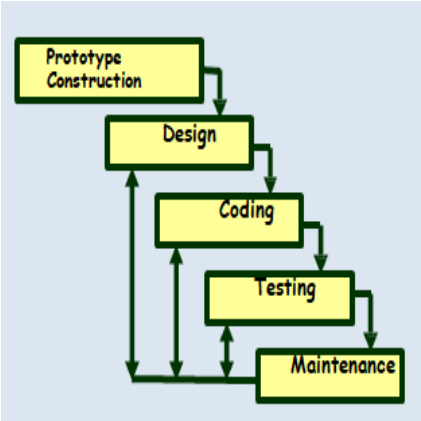
Difference between classic model and V-model

- In reality, there is no fundamental difference between the classic life cycle and the V-model. The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work

46

Prototype Model

- **A derivative of waterfall model.**
- **Before starting actual development,** A working prototype of the system should first be built.
- **A prototype is a toy implementation of a system:**
  - Limited functional capabilities,
  - Low reliability,
  - Inefficient performance.



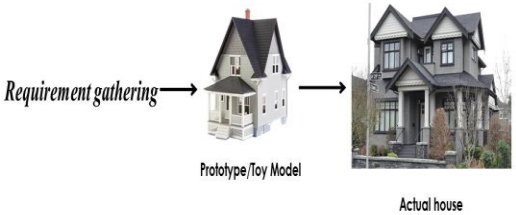
Reasons for Prototyping

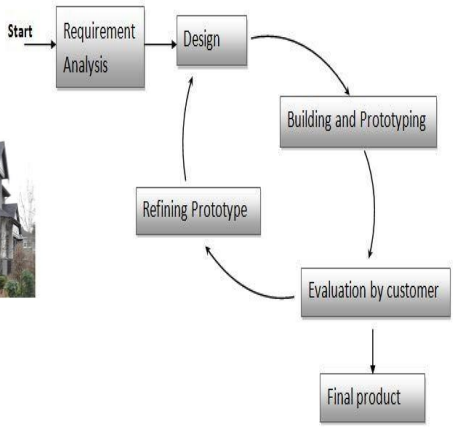
- **Learning by doing:** useful where requirements are only partially known.
- **Improved communication**
- **Improved user involvement**
- **Reduced need for documentation**
- **Reduced maintenance costs**
- **Illustrate to the customer:** input data formats, messages, reports, or interactive dialogs.
- **Examine technical issues associated with product development:**
  - Often major design decisions depend on issues like:
    - Response time of a hardware controller,
    - Efficiency of a sorting algorithm, etc.
- **Another reason for developing a prototype:** It is impossible to "get it right" the first time, We must plan to throw away the first version: If we want to develop a good software.

48

Prototype Model

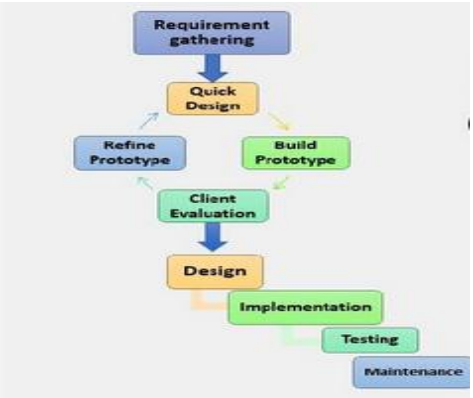
- **Start with approximate requirements.**
- **Carry out a quick design.**
- **Prototype is built using several short-cuts:**
  - Short-cuts might involve: Using inefficient, inaccurate, or dummy functions.
  - A table look-up rather than performing the actual





Prototype Model

- **The developed prototype is submitted to the customer for his evaluation:**
- Based on the user feedback, the prototype is refined.
- This cycle continues until the user approves the prototype.
- **The actual system is developed using the waterfall model.**
- **Requirements analysis and specification phase becomes redundant:**
  - Final working prototype (incorporating all user feedbacks) serves as an animated requirements specifications



Prototype Model

- **The developed prototype is submitted to the customer for his evaluation:**
- **Design and code for the prototype is usually thrown away:**
  - However, experience gathered from developing the prototype helps a great deal while developing the actual software.
- Even though construction of a working prototype model involves additional cost — overall development cost usually lower for:
  - Systems with unclear user requirements,
  - Systems with unresolved technical issues.
- Many user requirements get properly defined and technical issues get resolved:
  - These would have appeared later as change requests and resulted in incurring massive redesign costs.

51

Prototyping Models

When to use ?

- **Customers have** general **objectives of software** but **do not have detailed requirements** for functions & features.
- **Developers** are **not sure** about **efficiency of an algorithm** & **technical feasibilities**.

- It serves as a **mechanism** for **identifying software requirements**.
- Prototype can be serve as "**the first system**".
- Both stakeholders and software engineers like prototyping model
  - Users get feel for the actual system
  - Developers get to build something immediately

52

Prototyping Models

Problem areas

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

Advantages

- **Users** are actively **involved** in the **development.**
- Since in this methodology a working model of the system is provided, the **users get a better understanding** of the **system** being developed.
- **Errors** can be **detected** much **earlier**

53

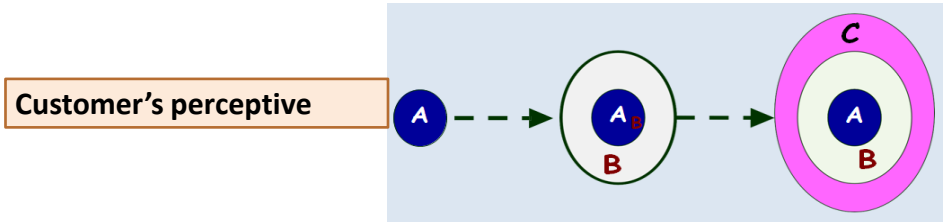
Incremental Models

- The incremental model **combines** elements of **linear** and **parallel** process flows.
- This model applies linear sequence in a iterative manner.
- Initially **core working product** is **delivered**.
- **Each** linear **sequence** produces deliverable **"increments"** of the software.
- e.g. **word-processing software** developed **using** the **incremental model**
  - It might deliver basic file management, editing and document production functions in the first increment
  - more complex editing in the second increment;
  - spelling and grammar checking in the third increment; and
  - advanced page layout capability in the fourth increment.

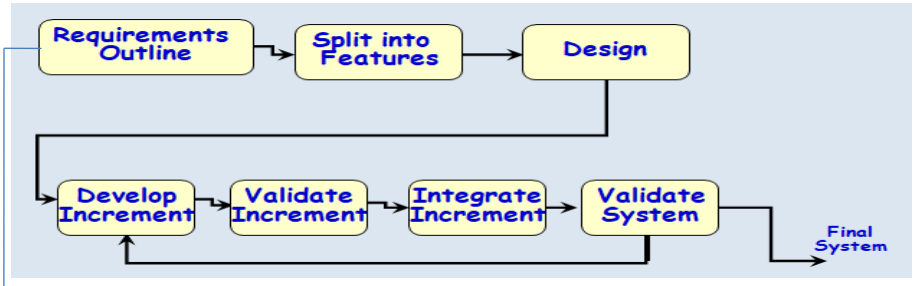
54

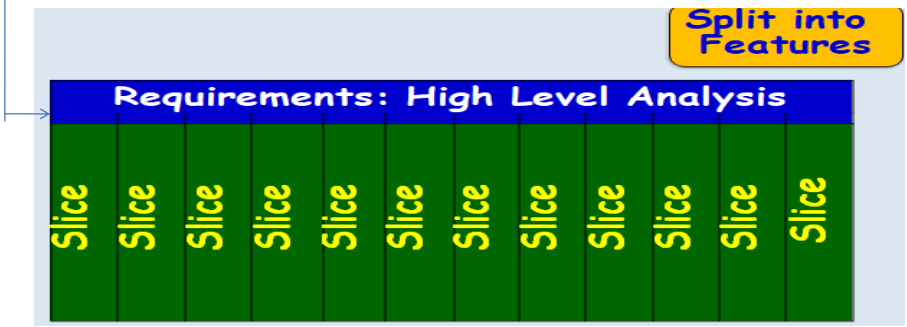
Incremental and Iterative Models

- **Key characteristics**
  - Builds system incrementally
  - Consists of a planned number of iterations
  - Each iteration produces a working program
- **Benefits :** Facilitates and manages changes
- **Foundation of agile techniques and the basis for**
  - Rational Unified Process (RUP)
  - Extreme Programming (XP)



Incremental Models

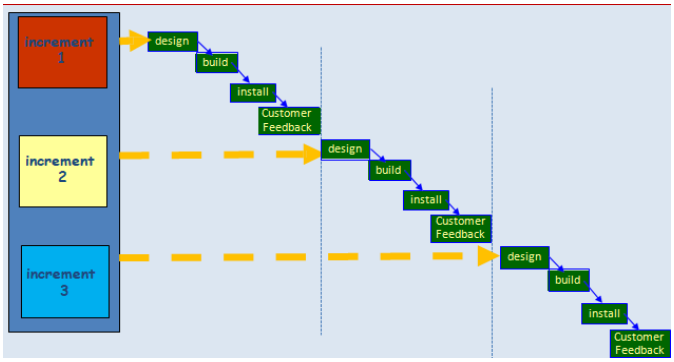




Incremental and Iterative Models

- **Waterfall: single release**
- **Iterative: many releases (increments)**
  - First increment: core functionality
  - Successive increments: add/fix functionality
  - Final increment: the complete product
- **Each iteration: a short mini-project with a separate Lifecycle**
  e.g., waterfall

Incremental delivery



Incremental Models

When to use ?

- When the **requirements** of the **complete** system are clearly **defined** and understood but **staffing is unavailable** for a **complete implementation** by the business deadline.

Advantage

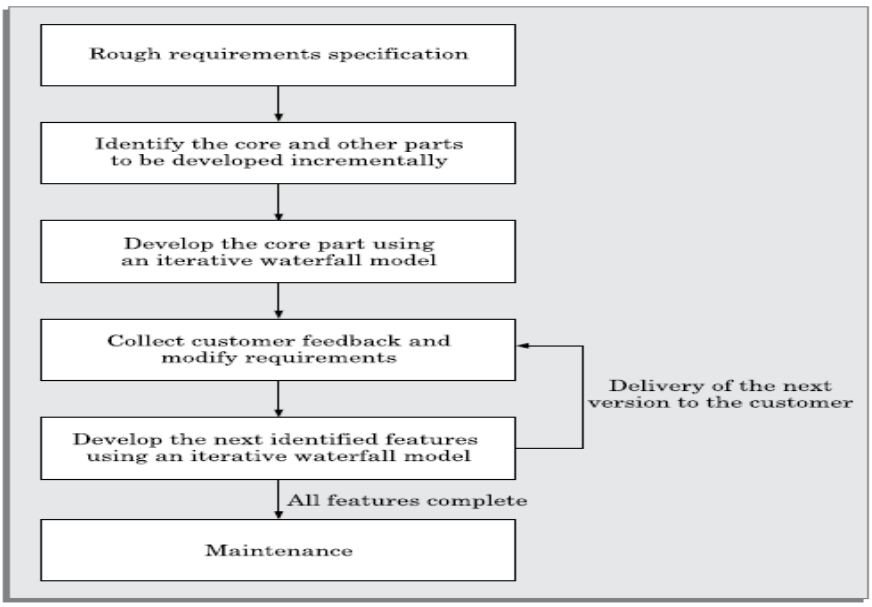
- Generates **working software quickly** and early during the software life cycle.
- It is **easier to test** and debug during a smaller iteration.
- **Customer** can **respond** to each built.
- **Lowers initial** delivery **cost**.
- **Easier** to **manage risk** because risky pieces are identified and handled during iteration.

Evolutionary Models

- Software system evolves over time as requirements often change as development proceeds. Thus, a straight line to a complete end product is not possible.
- However, a limited version must be delivered to meet competitive pressure.
- When a set of **core product** or system requirements is **well understood** but the **details of product** or system extensions have **yet to be defined**.
- In this situation there is **a need of process model** which specially designed to accommodate **product** that **evolve with time**.
- **Evolutionary Process Models** are specially meant for that which produce an increasingly more complete version of the software with each iteration.
- Evolutionary Models are **iterative**.

59

Evolutionary Models



Evolutionary Models

- **Recognizes the reality of changing requirements**
  - **Capers Jones's research on 8000 projects: 40% of final requirements arrived after development had already begun**
- **Promotes early risk mitigation:**
  - Breaks down the system into mini-projects and focuses on the riskier issues first.
  - "plan a little, design a little, and code a little"
- **Encourages all development participants to be involved earlier on,:** End users, Testers, integrators, and technical writers

61

Evolutionary Models

- **First develop the core modules of the software.**
- **The initial skeletal software is refined into increasing levels of capability: (Iterations) :** By adding new functionalities in successive versions.
- **Software developed over several "mini waterfalls".**
- **The result of a single iteration:**
  - Ends with delivery of some tangible code
  - An incremental improvement to the software – leads to evolutionary development
- **Outcome of each iteration: tested, integrated, executable system**
- **Iteration length is short and fixed :** Usually between 2 and 6 weeks.
  - Development takes many iterations (for example: 10-15)

62

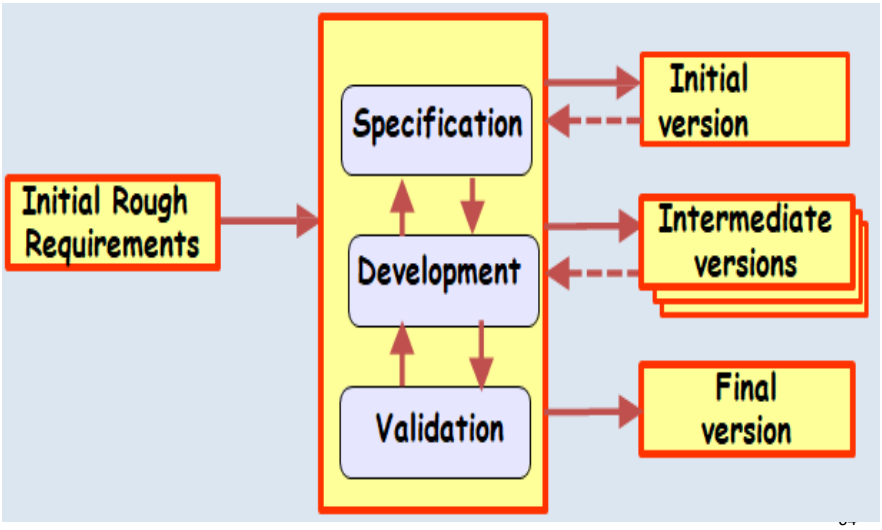
Evolutionary Models

- **Does not "freeze" requirements and then conservatively design :**
  - Opportunity exists to modify requirements as well as the design…
- **Successive versions:**
  - Functioning systems capable of performing some useful work.
  - A new release may include new functionality: Also existing functionality in the current release might have been enhanced.

63

Evolutionary Models

- **Evolves an initial implementation with user feedback:**
  - Multiple versions until the final version.



Advantages of Evolutionary Models

- **Users get a chance to experiment with a partially developed system:** Much before the full working version is released,
- **Helps finding exact user requirements:** Software more likely to meet exact user requirements.
- **Core modules get tested thoroughly:** Reduces chances of errors in final delivered software.
- **Better management of complexity by developing one increment at a time.**
- **Better management of changing requirements.**
- **Can get customer feedback and incorporate them much more efficiently:** As compared when customer feedbacks come only after the development work is complete.
- **Training can start on an earlier release :** customer feedback taken into account
- **Frequent releases allow developers to fix unanticipated problems quicker.**

65

Problems with Evolutionary Models

- **The process is intangible:** No regular, well-defined deliverables.
- **The process is unpredictable:** Hard to manage, e.g., scheduling, workforce allocation, etc.
- **Systems are rather poorly structured:** Continual, unpredictable changes tend to degrade the software structure.
- **Systems may not even converge to a final version.**

66

RAD (Rapid Application Development) Models

- **Sometimes referred to as the rapid prototyping model.**
- **Major aims:**
  - Decrease the time taken and the cost incurred to develop software systems.
  - Facilitate accommodating change requests as early as possible:
    - Before large investments have been made in development and testing.
  - To reduce the communication gap between the customer and the developers.
- **A way to reduce development time and cost, and yet have flexibility to incorporate changes:**

> ✧ **Make only short term plans and make heavy reuse of existing code.**

RAD Methodology

- **Plans are made for one increment at a time.**
  - **The time planned for each iteration is called a time box.**
- **Each iteration (increment): Enhances the implemented functionality of the application a little.**
- **During each iteration,**
  - A quick-and-dirty prototype-style software for some selected functionality is developed.
  - The customer evaluates the prototype and gives his feedback.
  - The prototype is refined based on the customer feedback.

How Does RAD Facilitate Faster Development?

- **RAD achieves fast creation of working prototypes :** Through use of specialized tools.
- **These specialized tools usually support the following features:**
  - Visual style of development.
  - Use of reusable components.
  - Use of standard APIs (Application Program Interfaces).

69

For which Applications is RAD Suitable?

- **Customized product developed for one or two customers only**
- **Performance and reliability are not critical.**
- **The system can be split into several independent modules.**

For which Applications is RAD Unsuitable?

- **Few plug-in components are available**
- **High performance or reliability required**
- **No precedence for similar products exists**
- **The system cannot be modularized.**

70

Prototyping versus RAD

- **In prototyping model:**
  - The developed prototype is primarily used to gain insights into the solution
  - Choose between alternatives
  - Elicit customer feedback.
  - **The developed prototype:** Usually thrown away.
- **In contrast:** In RAD the developed prototype evolves into deliverable software.
  - **RAD leads to faster development compared to traditional models:** However, the quality and reliability would possibly be poorer.

71

RAD versus Iterative Waterfall Model

- **In the iterative waterfall model,** All product functionalities are developed together.
- **In the RAD model on the other hand,** Product functionalities are developed incrementally through heavy code and design reuse.
  - Customer feedback is obtained on the developed prototype after each iteration: Based on this the prototype is refined.
- **The iterative waterfall model:** Does not facilitate accommodating requirement change requests.
- **Iterative waterfall model does have some important advantages:**
  - Use of the iterative waterfall model leads to production of good documentation.
  - Also, the developed software usually has better quality and reliability than that developed using RAD.

72

RAD versus Evolutionary Model

- **Incremental development:** Occurs in both evolutionary and RAD models.
- **However, in RAD:**
  - Each increment is a quick and dirty prototype,
  - Whereas in the evolutionary model each increment is systematically developed using the iterative waterfall model.
- **Also, RAD develops software in shorter increments:**
  - The incremental functionalities are fairly large in the evolutionary model.

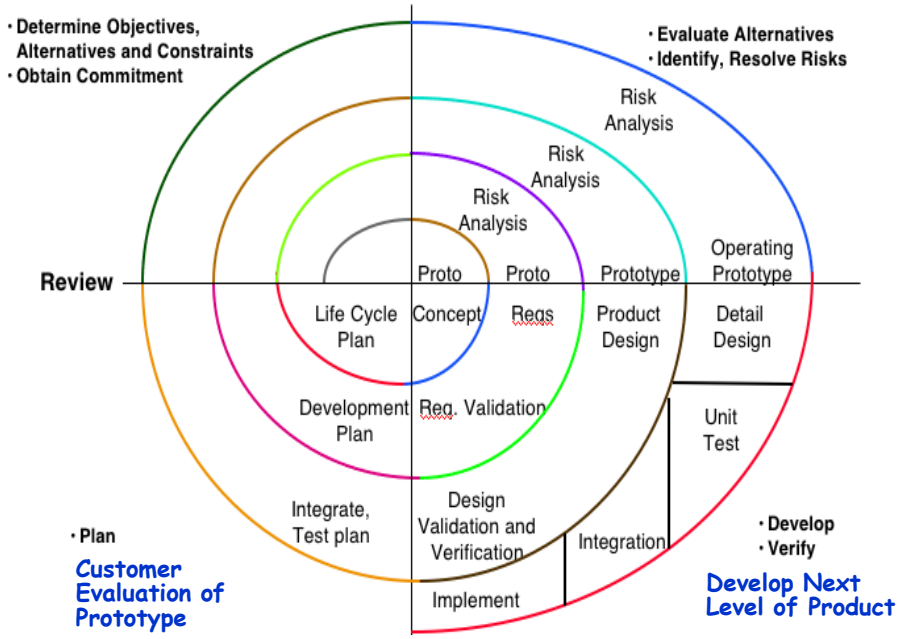
73

Spiral Model

- **Proposed by Boehm in 1988.**
- **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.
- **There are no fixed phases in this model, the phases shown in the figure are just examples.**
- **The team must decide:** how to structure the project into phases.
- **Start work using some generic model:** add extra phases for specific projects or when problems are identified during a project.
- **Each loop in the spiral is split into four sectors (quadrants).**

74

Spiral Model



Objective Setting (First Quadrant)

- **Identify objectives of the phase,**
- **Examine the risks associated with these objectives.**
  - **Risk:** Any adverse circumstance that might hamper successful completion of a software project.
- **Find alternate solutions possible.**

Risk Assessment and Reduction (Second Quadrant)

- **For each identified project risk,** a detailed analysis is carried out.
- **Steps are taken to reduce the risk.**
- **For example, if there is a risk that requirements are inappropriate:** A prototype system may be developed.

Development and Validation (Third quadrant)

- develop and validate the next level of the product.

Review and Planning (Fourth quadrant)

- review the results achieved so far with the customer and plan the next iteration around the spiral.

- **With each iteration around the spiral**: progressively more complete version of the software gets built.

77

Spiral Models

- It couples the **iterative** nature of prototyping with the controlled and systematic aspects of the **waterfall model** and is a **risk-driven** process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems.
- Two main distinguishing features:
  - one is cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk.
  - The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.
- A series of evolutionary releases are delivered. During the early iterations, the release might be a model or prototype. During later iterations, increasingly more complete version of the engineered system are produced.

78

Spiral Models

When to use :

- For development of **large scale / high-risk projects**.
- When costs and **risk evaluation is important**.
- Users are **unsure** of their **needs**.
- **Requirements** are **complex**.
- New product line.
- Significant (**considerable**) **changes** are expected.

Advantage

- High amount of risk analysis hence, **avoidance of Risk** is enhanced.
- **Strong approval** and **documentation** control.
- **Additional functionality** can be **added** at a later date.
- **Software** is **produced early** in the Software Life Cycle.

79

Spiral Models

Disadvantage

- Can be **a costly model** to use.
- Risk analysis **requires highly specific expertise**.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

80

A Comparison Of Different Life Cycle Models

Software lifecycle Model	Features	Advantages	Disadvantages
Classical waterfall model	• Has defined phases •Follows all phases in the mentioned order •The flow of phase execution cannot be broken •Iteration is not allowed	•Simple to use, understand and implement •Managed work	•No scope of risk management •New features cannot be added or modified in the software after the development has begun. •Error is detected at the end of every phase

81

A Comparison Of Different Life Cycle Models

Software lifecycle Model	Features	Advantages	Disadvantages
Iterative lifecycle model	•Enhanced classical waterfall model in which Iterations are allowed •Most commonly used	•Risk management is possible •Updates and modification can be made in any phase •Easy to understand and implement	•Risks cannot be handled efficiently •Developer's idle hours are too high

82

A Comparison Of Different Life Cycle Models

Software lifecycle Model	Features	Advantages	Disadvantages
Prototyping model	•The prototype of the model is made before the actual software •User Interface usually is much more attractive in this model	•It can be used where the user requirements are not defined. •Takes user feedback from time to time •User's satisfaction is a priority here	•Is costly •The communication that is required between the customer and the developer is not always possible to get •Takes a lot of time to develop

83

A Comparison Of Different Life Cycle Models

Software lifecycle Model	Features	Advantages	Disadvantages
Evolutionary model	•Software is developed through different modules in an incremental manner •This model deals with the different versions of the software •Each version is an enhanced version of the previous one	•Large projects can be developed efficiently through this model •Every version is capable of to fully function the mentioned functionalities	•Is suitable only for large projects •Takes time to develop •Module integration is difficult

84

A Comparison Of Different Life Cycle Models

Software lifecycle Model	Features	Advantages	Disadvantages
Spiral model	•The phases are divided in the form of loops •Further, the loops are divided into four quadrants •Each block of this model contains a set of activities that the software performs	•Most versatile model •Able to manage almost every type of risk •Complex projects can be created using this model	•Is complex to understand and implement •Not suitable for ordinary software models •Costly to develop •Time consuming

Choosing the Right Life Cycle Model for a Project

- **Software characteristics should be developed**
  - The life cycle model used is primarily determined by the sort of software being produced.
  - The agile paradigm is preferred for small service projects.
  - The Iterative Waterfall paradigm, on the other hand, may be preferable for product and embedded development.
  - An object-oriented project may be developed using the evolutionary approach.
  - The project's user interface is mostly created using a prototyping technique.

Choosing the Right Life Cycle Model for a Project

- **Development team characteristics**
  - The skill level of team members is a key aspect in determining which life cycle model to utilize.
  - Even embedded software may be produced utilizing the Iterative Waterfall technique if the development team has prior expertise with comparable software.
  - Even a basic data processing application may need a prototype methodology if the development team is completely inexperienced.

87

Choosing the Right Life Cycle Model for a Project

- **Project risk -**
  - If the risks are limited and can be predicted at the outset of the project, a prototyping approach might be effective.
  - The spiral model is the appropriate model to utilize if the risks are difficult to predict at the start of the project but are anticipated to rise as the development progresses.

88

Choosing the Right Life Cycle Model for a Project

- **Client characteristics –**
  - If the customer is unfamiliar with computers, the requirements are likely to change often since developing full, consistent, and clear needs would be challenging.
  - As a result, a prototyping model may be required to reduce customer change requests in the future.
  - The customer's trust in the development team is initially great. Because no operational software is evident throughout the long development phase, client trust tends to wane.
  - As a result, the evolutionary paradigm is beneficial since customers may experience partly functioning software far sooner than they can experience fully functional software.
  - Another benefit of the evolutionary approach is that it alleviates the customer's anxiety about learning a new system.

89