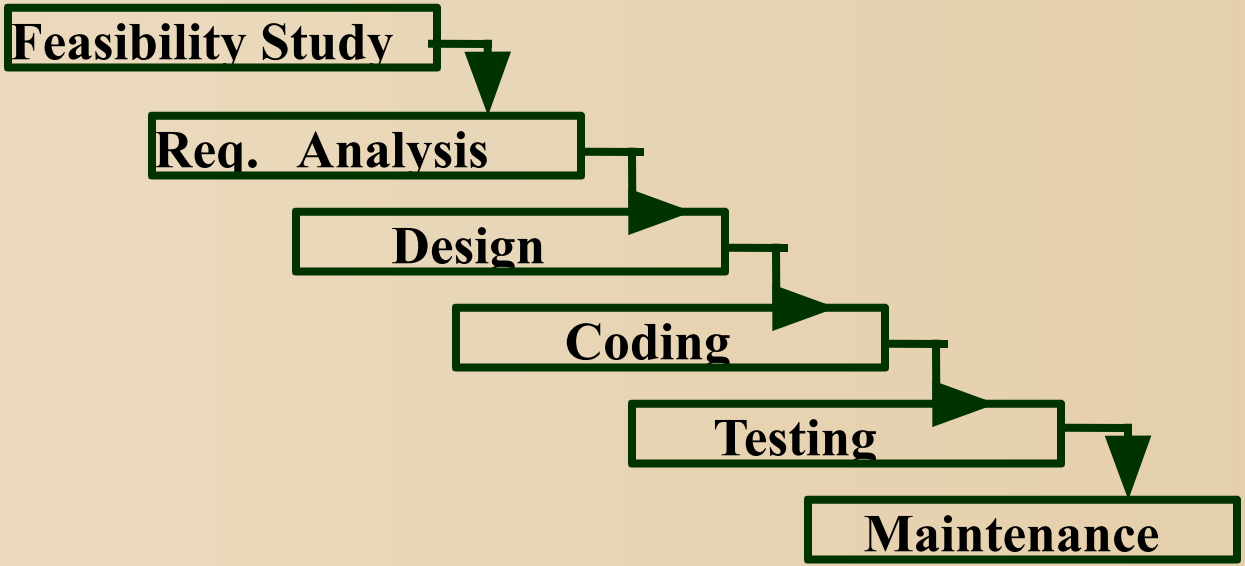
Process Models

Classical Waterfall Model

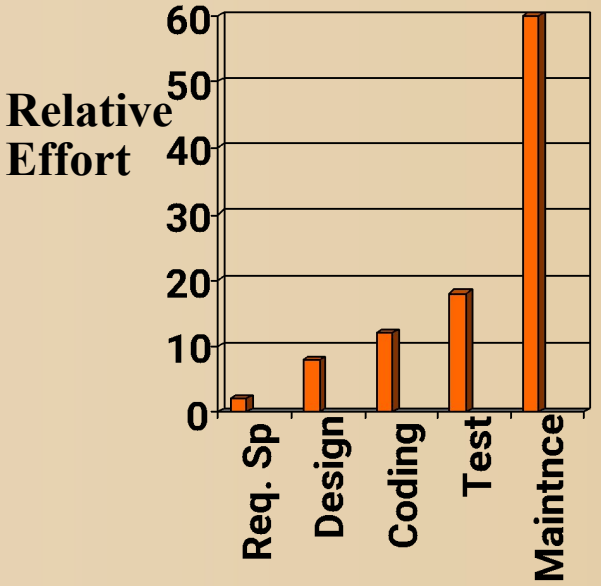
- **Classical waterfall model divides life cycle into phases:**
  - **feasibility study,**
  - **requirements analysis and specification,**
  - **design,**
  - **coding and unit testing,**
  - **integration and system testing,**
  - **maintenance.**

Classical Waterfall Model



Relative Effort for Phases

- **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.**



Classical Waterfall Model (CONT.)

- **Most organizations usually define:**
  - standards on the outputs (deliverables) produced at the end of every phase
  - entry and exit criteria for every phase.

Feasibility Study

- **Main aim of feasibility study:determine whether developing the product**
  - **financially worthwhile**
  - **technically feasible.**
- **First roughly understand what the customer wants:**
  - **different 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.**

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.**

Activities during Feasibility Study

- **Perform a cost/benefit analysis:**
  - **to determine which solution is the best.**
  - **you may determine that none of the solutions is feasible due to:**
    - **high cost,**
    - **resource constraints,**
    - **technical reasons.**

Case study - **Galaxy Mining Company Ltd. (GMC)**

Special Provident Fund (SPF) Software Project

Background : ~50 mine sites across 8 Indian states, Large workforce of miners, Mining is a high-risk profession, The main objective of having the special provident fund (SPF) would be quickly distribute some compensation before the standard provident amount is paid.

Objective:

- Automate SPF record maintenance
- Speedy settlement of compensation claims
- Reduce bookkeeping manpower
- Ensure timely distribution of SPF payouts

Budget & Vendor:

- **Budget:** ₹1 million
- **Vendor:** Adventure Software Inc.
- Feasibility study by vendor's project manager
- Discussions with GMC management & field PF officers

Case study - **Galaxy Mining Company Ltd. (GMC)**
Special Provident Fund (SPF) Software Project

Proposed Approaches

1. **Central Databases -** Satellite connection to all mine sites
2. **Local Databases**
   - Each site keeps local records
   - Periodic updates to central database via dial-up

Feasibility Findings

- Approach 2 is **more affordable** and **fault-tolerant**
- Local operations continue if central link fails
- Database functionality, UI, and communication modules analyzed
- Cost and technology proven feasible

Final Decision: Adopt local databases with periodic central updates, Daily or hourly synchronization acceptable, Financially and technically approved by GMC management.

Requirements Analysis and Specification

- **Aim of this phase:**
  - **understand the exact requirements of the customer,**
  - **document them properly.**
- **Consists of two distinct activities:**
  - **requirements gathering and analysis**
  - **requirements specification.**

Goals of Requirements Analysis

- **Collect all related data from the customer:**
  - **analyze the collected data to clearly understand what the customer wants,**
  - **find out any inconsistencies and incompleteness in the requirements,**
  - **resolve all inconsistencies and incompleteness.**

Requirements Gathering

- **Gathering relevant data:**
  - **usually collected from the end-users through interviews and discussions.**
  - **For example, for a business accounting software:**
    - **interview all the accountants of the organization to find out their requirements.**

Requirements Analysis (CONT.)

- **The data you initially collect from the users:**
  - **would usually contain several contradictions and ambiguities:**
  - **each user typically has only a partial and incomplete view of the system.**

Requirements Analysis (CONT.)

- **Ambiguities and contradictions:**
  - **must be identified**
  - **resolved by discussions with the customers.**
- **Next, requirements are organized:**
  - **into a Software Requirements Specification (SRS) document.**

Requirements Analysis (CONT.)

- **Engineers doing requirements analysis and specification:**
  - **are designated as analysts.**

Design

- **Design phase transforms requirements specification:**
  - **into a form suitable for implementation in some programming language.**

Design

- **In technical terms:**
  - **during design phase, software architecture is derived from the SRS document.**
- **Two design approaches:**
  - **traditional approach (Procedural approach),**
  - **object oriented approach.**

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,**
    - **Departments, etc.**

Object Oriented Design (CONT.)

- **Object structure**
  - **further refined to obtain the detailed design.**
- **OOD has several advantages:**
  - **lower development effort,**
  - **lower development time,**
  - **better maintainability.**

Implementation

- **Purpose of implementation phase (aka coding and unit testing phase):**
  - **translate software design into source code.**

Implementation

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

Implementation (CONT.)

- **The purpose of unit testing:**
  - **test if individual modules work correctly.**
- **The end product of implementation phase:**
  - **a set of program modules that have been tested individually.**

Integration and System Testing

- **Different modules are integrated in a planned manner:**
  - **modules are almost never integrated in one shot.**
  - **Normally integration is carried out through a number of steps.**
- **During each integration step,**
  - **the partially integrated system is tested.**

Integration and System Testing

System Testing

- **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.

Maintenance

- **Maintenance of any software product:**
  - requires much more effort than the effort to develop the product itself.
  - development effort to maintenance effort is typically 40:60.

Maintenance (CONT.)

- **Corrective maintenance:**
  - **Correct errors which were not discovered during the product development phases.**
- **Perfective maintenance:**
  - **Improve implementation of the system**
  - **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.**

Case study 1 - waterfall

✅ **Scenario Where Waterfall Is Most Suitable**

Project:

Library Management System for a University

Context:

The university wants a centralized system to handle book cataloguing, student borrowing/returning, and late-fee calculation. The administration has clearly documented requirements after years of using a manual process.

Case study 1 - waterfall

✅**Why It Fits Waterfall:**

1. **Stable Requirements**: All functions (book search, issue/return, fee calculation) are well-understood and unlikely to change.
2. **Regulatory Compliance**:The university requires detailed documentation for audits and funding approvals.
3. **Low Technological Risk**:The technology stack (web application, relational database) is well established.
4. **Predictable Timeline and Budget**:Funding and staffing are fixed; management wants a step-by-step, sign-off approach.

Execution:

- **Requirements**: Gather detailed specs from librarians and administrators.
- **Design**: Database schema, UI mockups, workflow diagrams.
- **Implementation**: Developers code modules for catalog, user management, and reporting.
- **Testing**: System test with dummy data, user acceptance by librarians.
- **Deployment & Maintenance**: Install on campus servers; provide yearly updates.

Case study 2 - waterfall

❌ **Scenario Where Waterfall Is NOT Suitable**

Project:

Social Fitness Mobile App for Start-up

Context:

A start-up wants to build an app that lets users track workouts, share progress, and challenge friends.

- The product concept is still evolving.
- User feedback will heavily influence the features.
- Market competition is fast-moving.

Case study 2 - waterfall

Why Waterfall Fails:

1. **Evolving Requirements**:
   - Early testers might request integrations with wearable devices or gamification features after initial release.
2. **Need for Rapid Iteration**:
   - Start-ups rely on frequent pivots and quick prototypes to find product–market fit.
3. **High Uncertainty**:
   - UI/UX experimentation and changing trends make upfront design rigid and risky.

Preferred Alternative:

- **Agile or Scrum**: Short sprints, continuous user feedback, and iterative releases let the team adapt quickly to market and customer input.

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.**

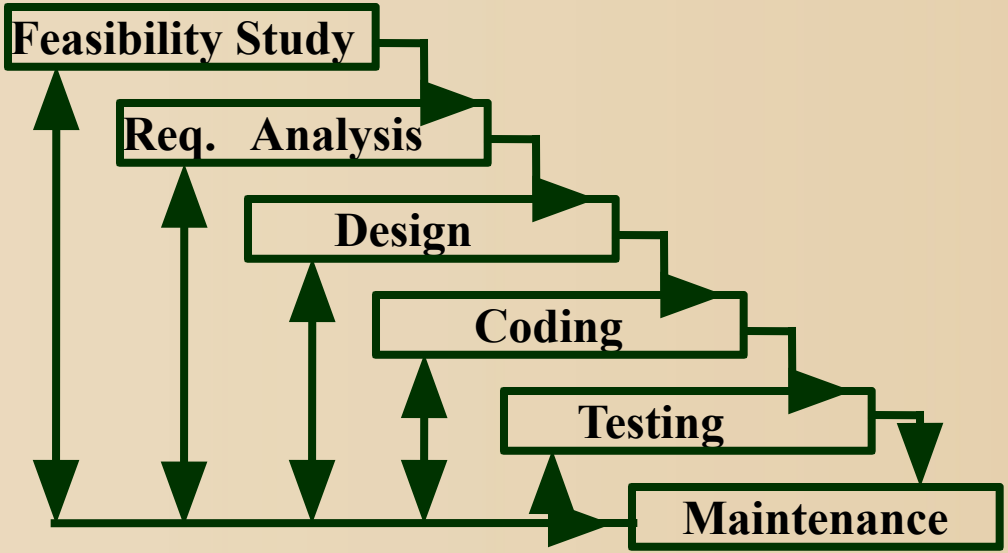
Iterative Waterfall Model (CONT.)

- **Defects usually get detected much later in the life cycle:**
  - For example, a design defect might go unnoticed till the coding or testing phase.

Iterative Waterfall Model (CONT.)

- **Once a defect is detected:**
  - we need to go back to the phase where it was introduced
  - redo some of the work done during that and all subsequent phases.
- **Therefore we need feedback paths in the classical waterfall model.**

Iterative Waterfall Model (CONT.)



Iterative Waterfall Model (CONT.)

- **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.

Phase containment of errors

- **Reason:** rework must be carried out not only to the design but also to code and test phases.
- **The principle of detecting errors as close to its point of introduction as possible:**
  - **is known as phase containment of errors .**
- **Iterative waterfall model is by far the most widely used model.**
  - **Almost every other model is derived from the waterfall model.**

Classical Waterfall Model (CONT.)

- **Irrespective of the life cycle model actually followed:**
  - **the documents should reflect a classical waterfall model of development,**
  - **comprehension of the documents is facilitated.**

Situations where most appropriate:

- Project has clear objectives and solution.
- Pressure does not exist for immediate implementation.
- Project requirements can be stated unambiguously and comprehensively.
- Project requirements are stable or unchanging during the system development life cycle.
- User community is fully knowledgeable in the business and application.
- Team members are experienced in developing similar types of projects.

Situations where least appropriate

- Large projects where the requirements are not well understood or are changing for any reasons such as external changes, changing expectations, budget changes or rapidly changing technology.
- Web Information Systems (WIS) primarily due to the pressure of implementing a WIS project quickly; the continual evolution of the project requirements; the need for experienced, flexible team members drawn from multiple disciplines; and the inability to make assumptions regarding the users' knowledge level.
- Real-time systems.
- Event-driven systems.

Case study 1 - iterative waterfall

✅ **Scenario Where Iterative Waterfall *Is* a Good Fit**

Project: Hospital Electronic Health Record (EHR) Enhancement

Context - A regional hospital is upgrading its existing EHR to add:

- A new patient portal
- Enhanced medication tracking
- Integration with a government reporting API
- The core patient-record functionality already exists. Regulations (HIPAA, national health data standards) mean **compliance and documentation are mandatory**, but there's a **moderate chance of evolving medical reporting rules** during the project.

Case study 1 - iterative waterfall

Why Iterative Waterfall Fits

1. **Stable Core Requirements**: The fundamental EHR structure is well defined.
2. **Regulatory Compliance + Feedback Loops**: Documentation and sign-off at each stage are mandatory, but the iterative feedback allows the team to step back if new government API specs arrive mid-project.
3. **Moderate, Predictable Change**: Only certain modules (e.g., reporting) might need adjustments; no constant re-prioritization.

Execution Flow

- **Requirements**: Gather current needs from hospital IT and regulatory bodies.
- **Design**: Prepare architecture for portal and API integration.
- **Implementation**: Develop portal and reporting modules.
- **Testing**: Discover a change in government API format → loop back to **Design** to revise, then re-implement.
- **Deployment**: Final EHR rollout with compliance checks.

Case study 2 - iterative waterfall

❌ Scenario Where Iterative Waterfall Is *Not* Suitable

Project:

AI-Powered Personalized Learning App for Children

Context:

A start-up wants to build a cross-platform app that uses AI to adapt lessons based on a child's pace and interests.

- They plan to experiment with gamification, new AI models, and frequent UI changes.
- Market and parent feedback will drive rapid pivots.
- They aim for continuous weekly releases.

Prototyping 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 developing a prototype

- 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.

Prototyping Model (CONT.)

- **The third reason for developing a prototype is:**
  - **it is impossible to ``get it right'' the first time,**
  - **we must plan to throw away the first product**
    - **if we want to develop a good product.**

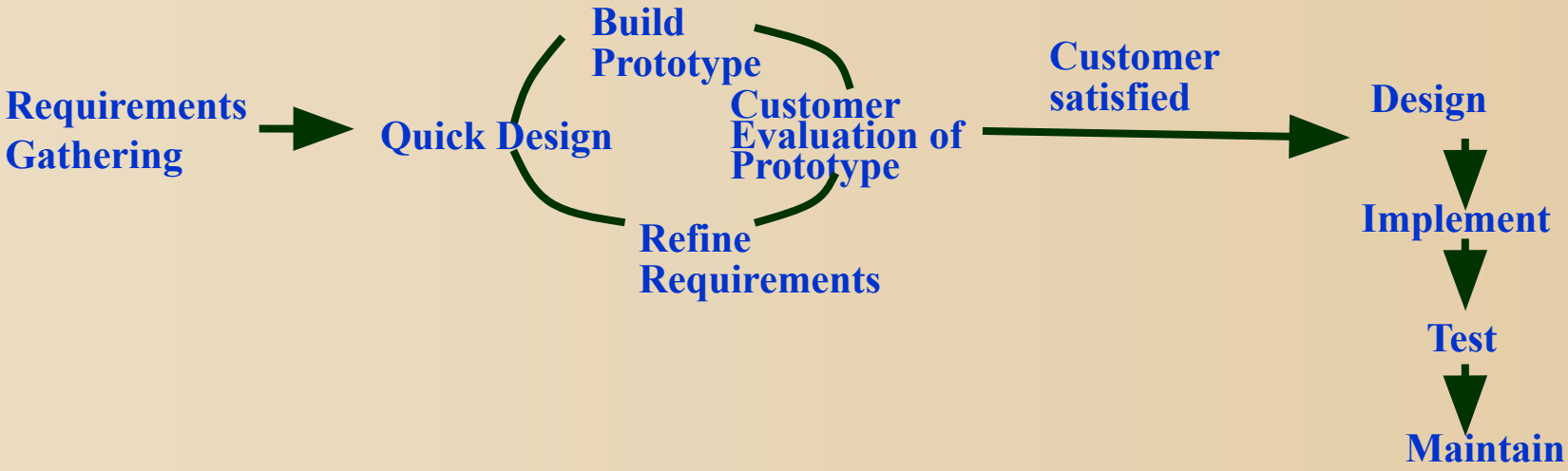
Prototyping Model (CONT.)

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

Prototyping Model (CONT.)

- **The developed prototype is submitted to the customer for his evaluation:**
  - Based on the user feedback, requirements are refined.
  - This cycle continues until the user approves the prototype.
- **The actual system is developed using the classical waterfall approach.**

Prototyping Model (CONT.)



Prototyping Model (CONT.)

- **Requirements analysis and specification phase becomes redundant:**
  - **final working prototype (with all user feedbacks incorporated) serves as an animated requirements specification.**
- **Design and code for the prototype is usually thrown away:**
  - **However, the experience gathered from developing the prototype helps a great deal while developing the actual product.**

Prototyping Model (CONT.)

- **Even though construction of a working prototype model involves additional cost --- overall development cost might be 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.

Situation where most appropriate

- Project is for development of an online system requiring extensive user dialog, or for a less well-defined expert and decision support system.
- Project objectives are unclear.
- Pressure exists for immediate implementation of something.
- Functional requirements may change frequently.
- User is not fully knowledgeable.

Case Study : School Event Registration Website

Project: Situation:

A local school wants a small website where parents can register children for annual events like sports day and science fair. Teachers know they need “online registration,” but they aren’t sure about:

- How the forms should look
- What details to collect
- How confirmation messages should appear

How Prototyping Helps:

1. **Quick Design & Prototype -** The developer creates a basic web page with a mock registration form and a sample confirmation page.
2. **User Feedback -** Teachers and a few parents try it and suggest changes (e.g., add a drop-down for event selection, add a payment option).
3. **Refine & Finalize -** Developer updates the prototype until everyone agrees.
4. **Full Development -** The final website is then built using the agreed design.

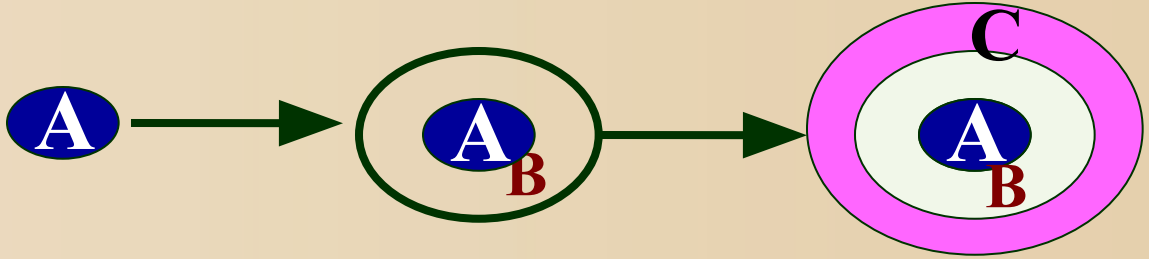
Evolutionary Model

- **Evolutionary model (aka successive versions or incremental model):**
  - **The system is broken down into several modules which can be incrementally implemented and delivered.**
- **First develop the core modules of the system.**
- **The initial product skeleton is refined into increasing levels of capability:**
  - **by adding new functionalities in successive versions.**

Evolutionary Model (CONT.)

- **Successive version of the product:**
  - **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.**

Evolutionary Model (CONT.)



Advantages of Evolutionary Model

- **Users get a chance to experiment with a partially developed system:**
  - **much before the full working version is released,**
- **Helps finding exact user requirements:**
  - **much before fully working system is developed.**
- **Core modules get tested thoroughly:**
  - **reduces chances of errors in final product.**

Disadvantages of Evolutionary Model

- **Often, difficult to subdivide problems into functional units:**
  - **which can be incrementally implemented and delivered.**
  - **evolutionary model is useful for very large problems, where it is easier to find modules for incremental implementation.**

Case Study : **Suitable for Evolutionary Model**

Project: **Enterprise-Scale E-Commerce Platform**

Context:

A large retail chain wants a comprehensive online marketplace with:

- Product catalog management
- Payment gateway integration
- Personalized recommendations
- Advanced analytics

The organization knows the **core requirements** (basic shopping, checkout, order tracking) but plans to **add features in stages** as the business grows.

Why Evolutionary Model Fits:

Incremental Delivery: Start with a minimum viable online store and gradually evolve to include AI recommendations, loyalty programs, and multi-currency support., **User Feedback Guides Growth:** Early customer behavior informs later releases., **Complex, Long-Term System:** Continuous enhancement over years matches the iterative nature of the model.

Evolutionary Model with Iteration

- **Many organizations use a combination of iterative and incremental development:**
  - **a new release may include new functionality**
  - **existing functionality from the current release may also have been modified.**

Evolutionary Model with iteration

Several advantages:

- **Training can start on an earlier release**
- **customer feedback taken into account**
- **Markets can be created:**
  - **for functionality that has never been offered.**
  - **Frequent releases allow developers to fix unanticipated problems quickly.**

Situations where most appropriate:

- Large projects where requirements are not well understood or are changing due to external changes, changing expectations, budget changes or rapidly changing technology.
- Web Information Systems (WIS) and event-driven systems.
- Leading-edge applications.

Situations where least appropriate:

- Very small projects of very short duration.
- Integration and architectural risks are very low.
- Highly interactive applications where the data for the project already exists (completely or in part), and the project largely comprises analysis or reporting of the data.

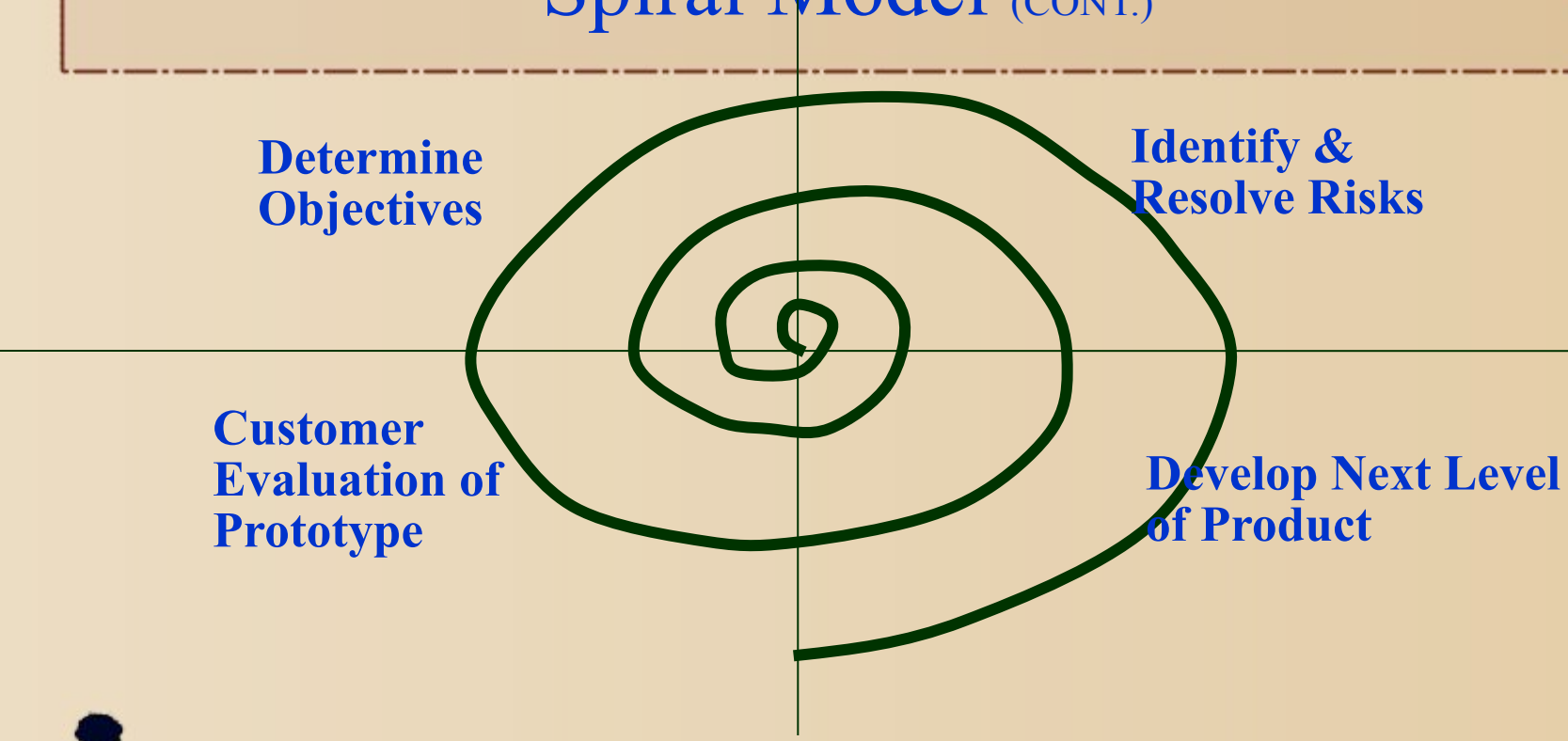
Spiral Model

- **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.**

Spiral Model (CONT.)

- **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).**

Spiral Model (CONT.)



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.**

Types of Risk in Project

The most common project risks are:

- Cost risk: typically escalation of project costs due to poor cost estimating accuracy and scope creep.
- Schedule risk: the risk that activities will take longer than expected. Slippages in schedule typically increase costs and, also, delay the receipt of project benefits, with a possible loss of competitive advantage.
- Performance risk: the risk that the project will fail to produce results consistent with project specifications.

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 the requirements are inappropriate:**
  - **a prototype system may be developed.**

Spiral Model (CONT.)

- **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.

Spiral Model as a meta model

- **Subsumes all discussed models:**
  - a single loop spiral represents waterfall model.
  - uses an evolutionary approach --
    - iterations through the spiral are evolutionary levels.
  - enables understanding and reacting to risks during each iteration along the spiral.
  - uses:
    - prototyping as a risk reduction mechanism
    - retains the step-wise approach of the waterfall model.

Situations where most appropriate:

- Real-time or safety-critical systems.
- Risk avoidance is a high priority.
- Minimizing resource consumption is not an absolute priority.
- Project manager is highly skilled and experienced.
- Requirement exists for strong approval and documentation control.
- Project might benefit from a mix of other development methodologies.
- A high degree of accuracy is essential.
- Implementation has priority over functionality, which can be a added in later versions.

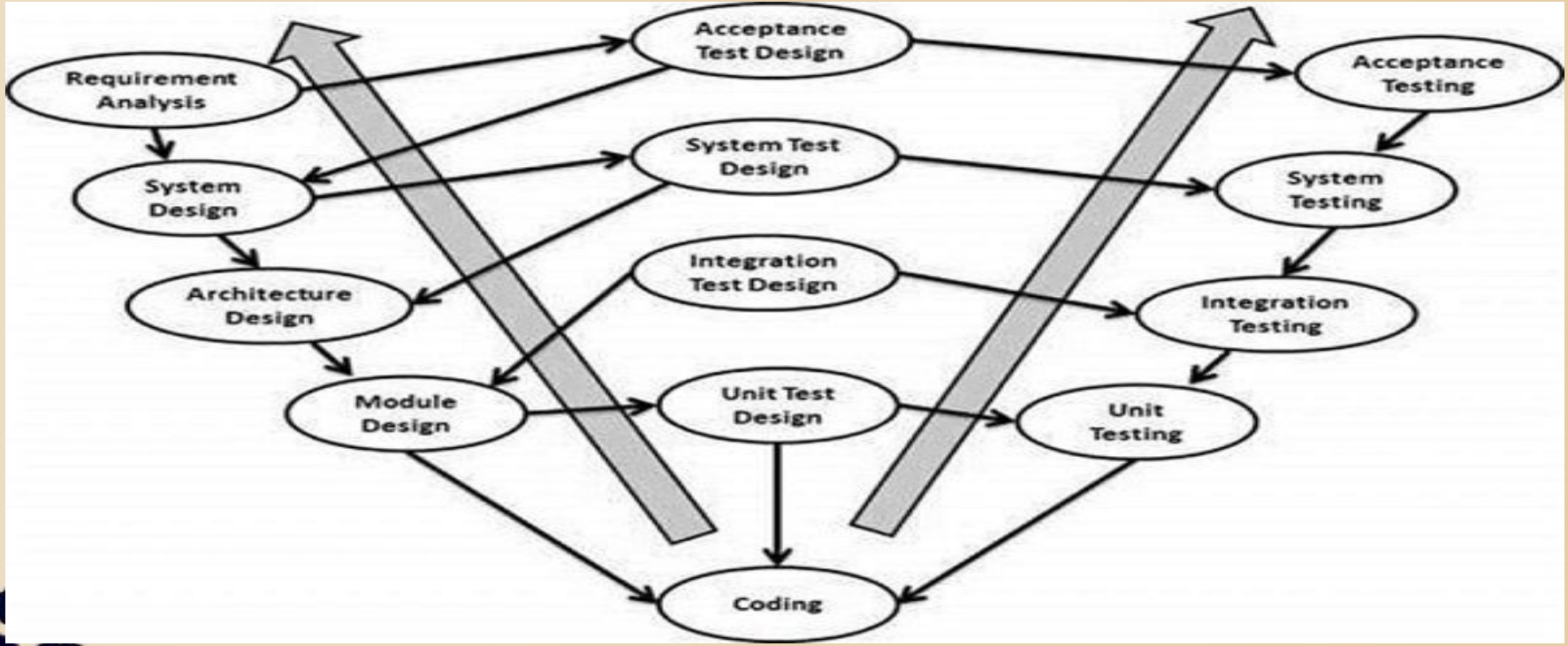
Situations where least appropriate:

- Risk avoidance is a low priority.
- A high degree of accuracy is not essential.
- Functionality has priority over implementation.
- Minimizing resource consumption is an absolute priority.

V-Model - Design

- The corresponding testing phase of the development phase is planned in parallel.
- there are Verification phases on one side of the ‘V’ and Validation phases on the other side.

V-Model - Design



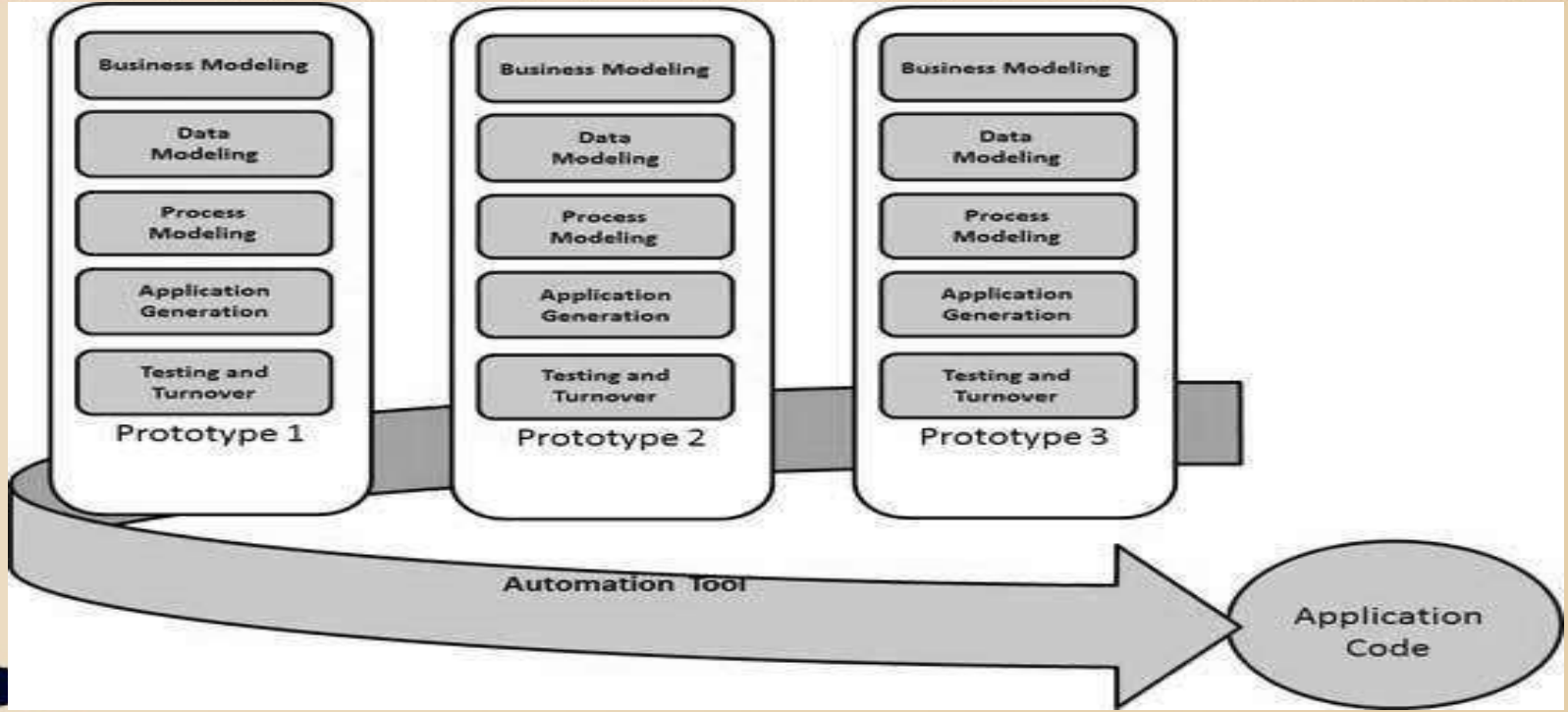
Rapid Application Development

- Functional modules are developed in parallel as prototypes.
- Integrated to make the complete product for faster product delivery.
- Follow iterative and incremental model.
- Prototypes developed are reusable.

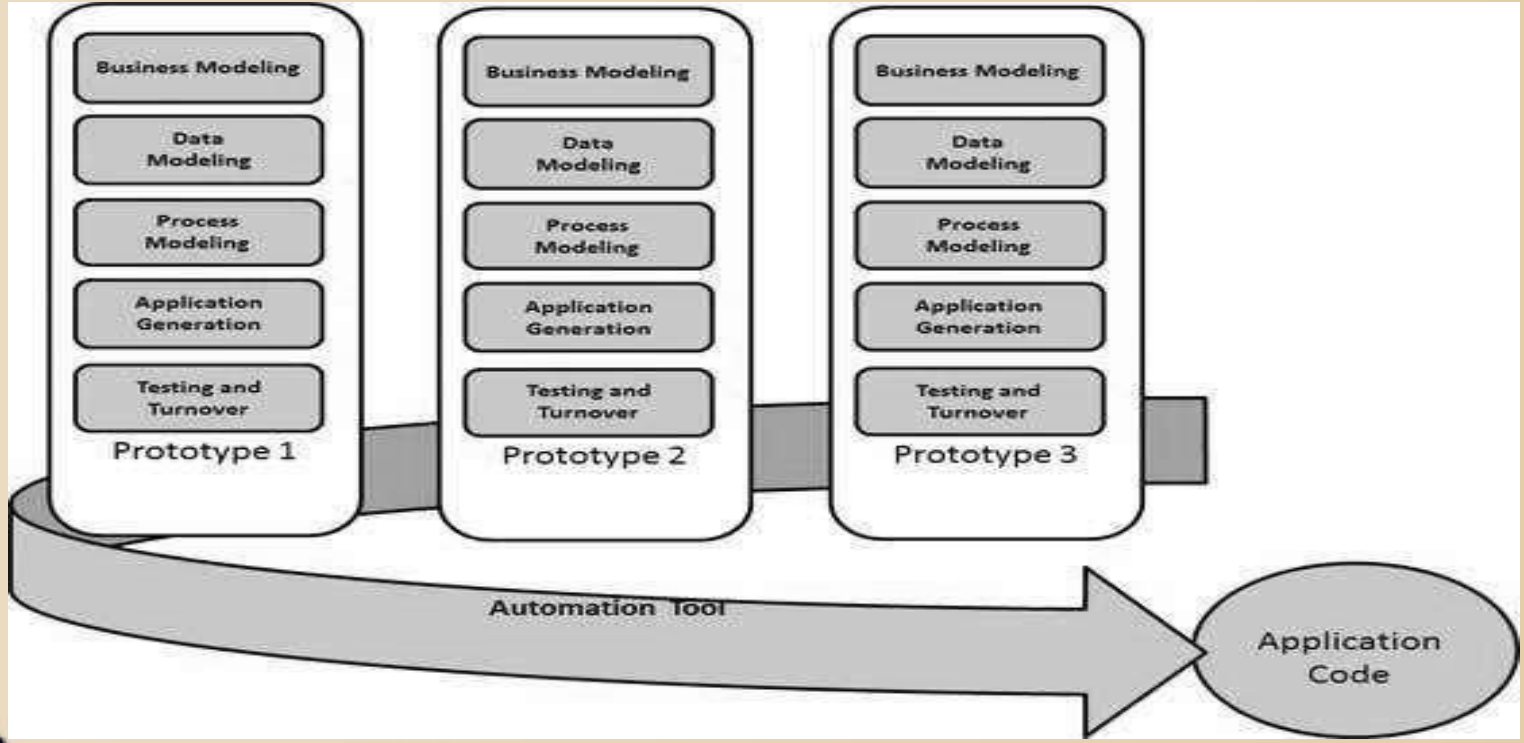
Rapid Application Development

- Quick Prototyping
- Iterative development
- Incremental Releases
- User involvement
- Time-boxing
- Parallel Processing

RAD Model


Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 3
Automation Tool
Application Code

RAD Model


Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 3
Automation Tool
Application Code

RAD

- The business model: designed in terms of flow of information and the distribution of information between various business channels.
- Data Modeling:Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business.
  - The attributes of all data sets is identified and defined.
  - The relation between these data objects are established.

Comparison of Different Life Cycle Models

- **Iterative waterfall model**
  - most widely used model.
  - But, suitable only for well-understood problems.
- **Prototype model is suitable for projects not well understood:**
  - **user requirements**
  - **technical aspects**

Comparison of Different Life Cycle Models

(CONT.)

- **Evolutionary model is suitable for large problems:**
  - can be decomposed into a set of modules that can be incrementally implemented,
  - incremental delivery of the system is acceptable to the customer.
- **The spiral model:**
  - suitable for development of technically challenging software products that are subject to several kinds of risks.

Mark the following as either True or False. Justify your answer.

Evolutionary life cycle model is ideally suited for development of very small software products typically requiring a few months of development effort.

.

Mark the following as either True or False. Justify your answer.

Evolutionary life cycle model is ideally suited for development of very small software products typically requiring a few months of development effort.

Ans.: - False.

Explanation: - The Evolutionary model is very useful for very large problems where it becomes easier to find modules for incremental implementation.

- Prototyping life cycle model is the most suitable one for undertaking a software development project susceptible to schedule slippage.

- Prototyping life cycle model is the most suitable one for undertaking a software development project susceptible to schedule slippage.
- Ans - False

Exercise
Which life cycle model would you follow

1) A well-understood data processing application.
2) A new software that would connect computers through satellite communication. (Assume that your team has no previous experience in developing satellite communication software.)
3) A software that would function as the controller of a telephone switching system.
4) A new library automation software that would link various libraries in the city.
5) An extremely large software that would provide, monitor, and control cellular communication among its subscribers using a set of revolving satellites.
6) A new text editor.
7) A compiler for a new language.
8) An object-oriented software development effort.
9) The graphical user interface part of a large software.

ANSWER

1. ITERATIVE WATER FALL MODEL
2. PROTOTYPE MODEL
3. PROTOTYPE/ITERATIVE WATER FALL MODEL
4. PROTOTYPE/ITERATIVE WATER FALL MODEL
5. SPIRAL/EVOLUTIONARY MODEL
6. PROTOTYPE MODEL
7. PROTOTYPE/ITERATIVE WATER FALL MODEL
8. EVOLUTIONARY MODEL
9. PROTOTYPE MODEL