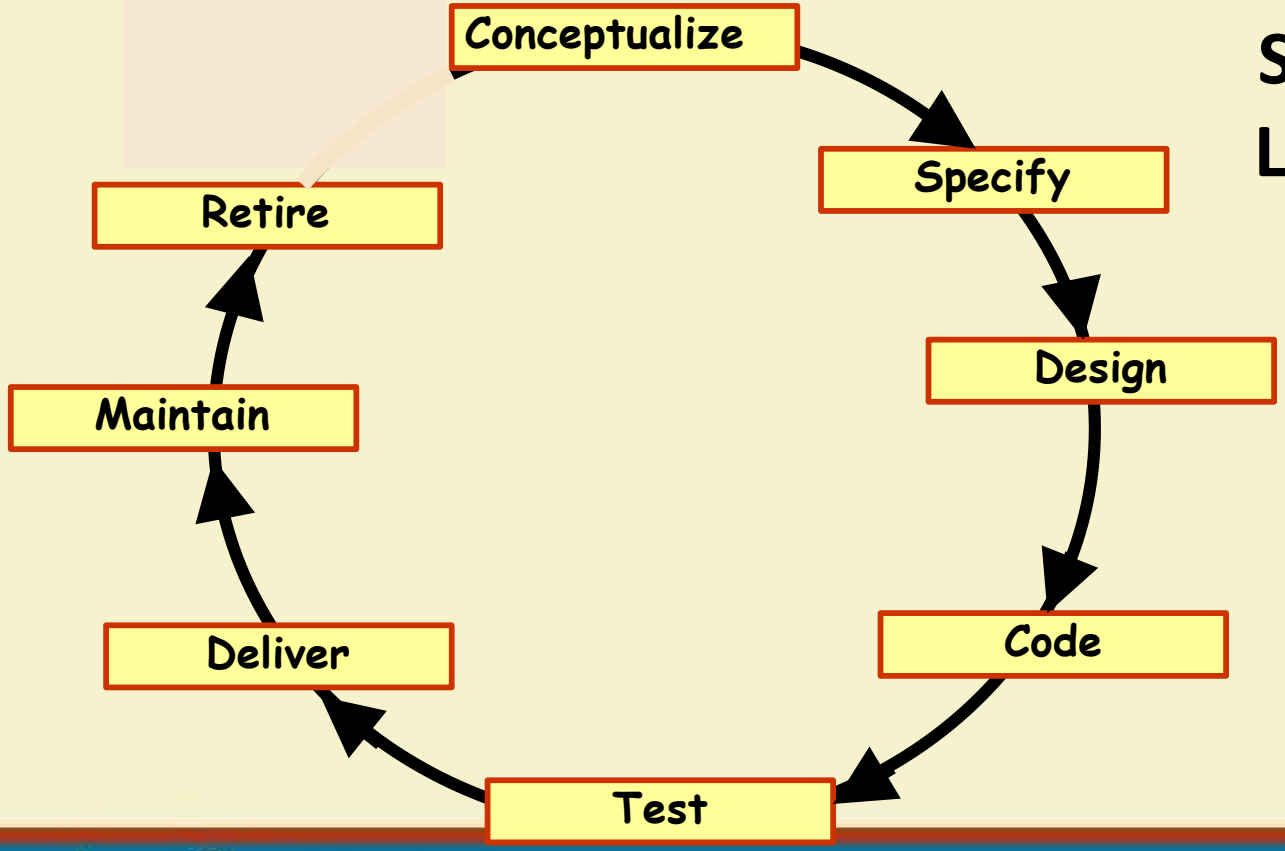
Life Cycle Models

Software Life Cycle



Life Cycle Model

- A software life cycle model (also process model or SDLC):
  - –A descriptive and diagrammatic model of software life cycle:
  - –Identifies all the activities undertaken during product development,
  - –Establishes a precedence ordering among the different activities,
  - –Divides life cycle into phases.

Life Cycle Model (CONT.)

- Each life cycle phase consists of several activities.
  - For example, the design stage might consist of:
    - structured analysis
    - structured design
    - Design review

Why Model Life Cycle?

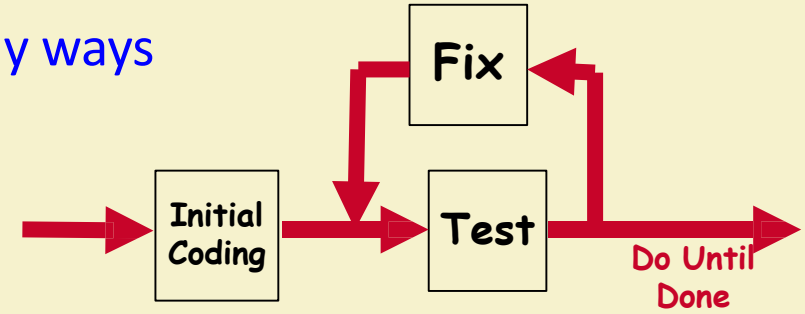
- A graphical and written description:
  - **Helps common understanding of activities among the software developers.**
  - **Helps to identify inconsistencies, redundancies, and omissions in the development process.**
  - **Helps in tailoring a process model for specific projects.**

Life Cycle Model (CONT.)

- The development team must identify a suitable life cycle model:
  - –and then adhere to it.
  - –Primary advantage of adhering to a life cycle model:
    - Helps development of software in a systematic and disciplined manner.

Life Cycle Model (CONT.)

- When a program is developed by a single programmer ---
  - The problem is within the grasp of an individual.
  - He has the freedom to decide his exact steps and still succeed --- called Exploratory model--- One can use it in many ways
  - Code➔Test ➔Design
  - Code➔Design ➔Test ➔ Change Code ➔
  - Specify ➔code ➔Design ➔Test ➔ etc.



Life Cycle Model (CONT.)

- When software is being developed by a team:
  - There must be a precise understanding among team members as to when to do what,
  - Otherwise, it would lead to chaos and project failure.

Life Cycle Model (CONT.)

- 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.

Phase Entry and Exit Criteria

- A life cycle model:

  – defines entry and exit criteria for every phase.

  – A phase is considered to be complete:

    - only when all its exit criteria are satisfied.

Life Cycle Model (CONT.)

- What is the phase exit criteria for the software requirements specification phase?
  - Software Requirements Specification (SRS) document is complete, reviewed, and approved by the customer.
- A phase can start:
  - Only if its phase-entry criteria have been satisfied.

Life Cycle Model: Milestones

- Milestones help software project managers:
  - –Track the progress of the project.
  - –Phase entry and exit are important milestones.

Life Cycle and Project Management

- When a life cycle model is followed:
  - –The project manager can at any time fairly accurately tell,
    - At which stage (e.g., design, code, test, etc.) the project is.

Project Management Without Life Cycle Model

- It becomes very difficult to track the progress of the project.
  - –The project manager would have to depend on the guesses of the team members.
- This usually leads to a problem:
  - –known as the **99% complete syndrome.**

Project Deliverables: Myth and Reality

Myth:

The only deliverable for a successful project is the working program.

Reality:

Documentation of all aspects of software development are needed to help in operation and maintenance.

Life Cycle Model (CONT.)

- 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

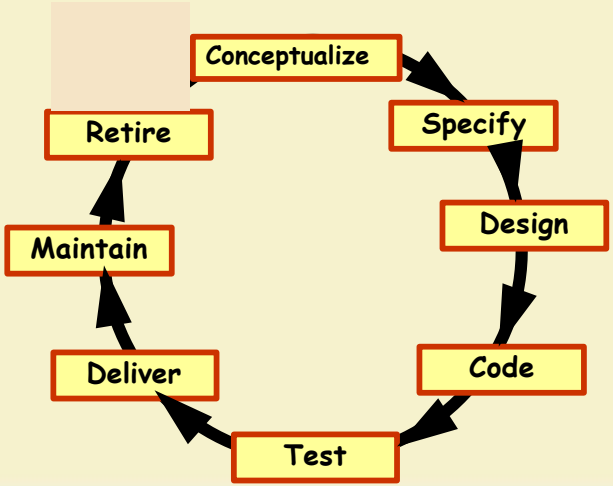
Traditional models

- Software life cycle (or 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.

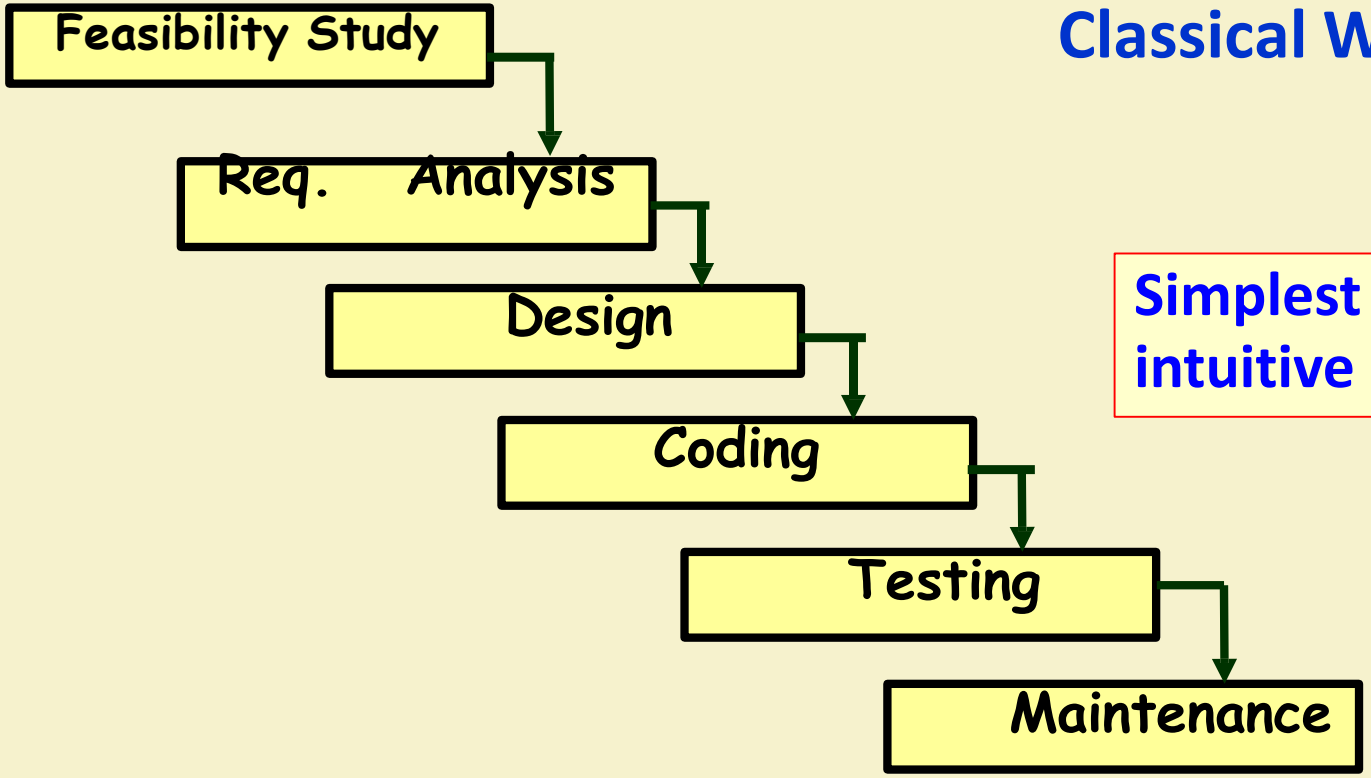
Software Life Cycle

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.



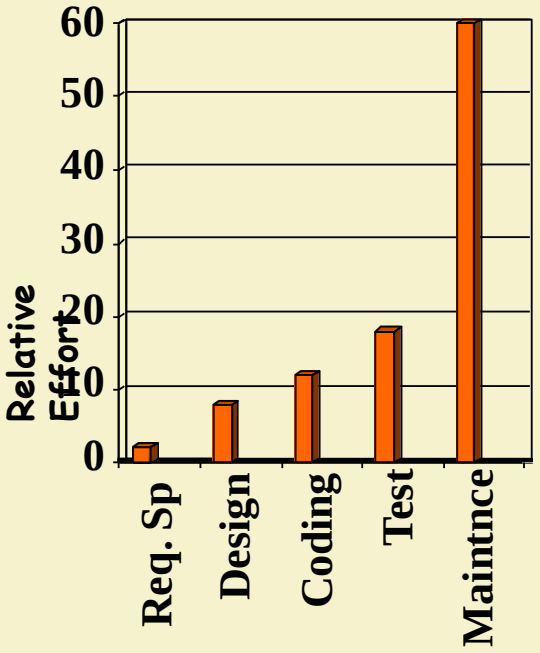
Classical Waterfall Model



Simplest and most intuitive

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.



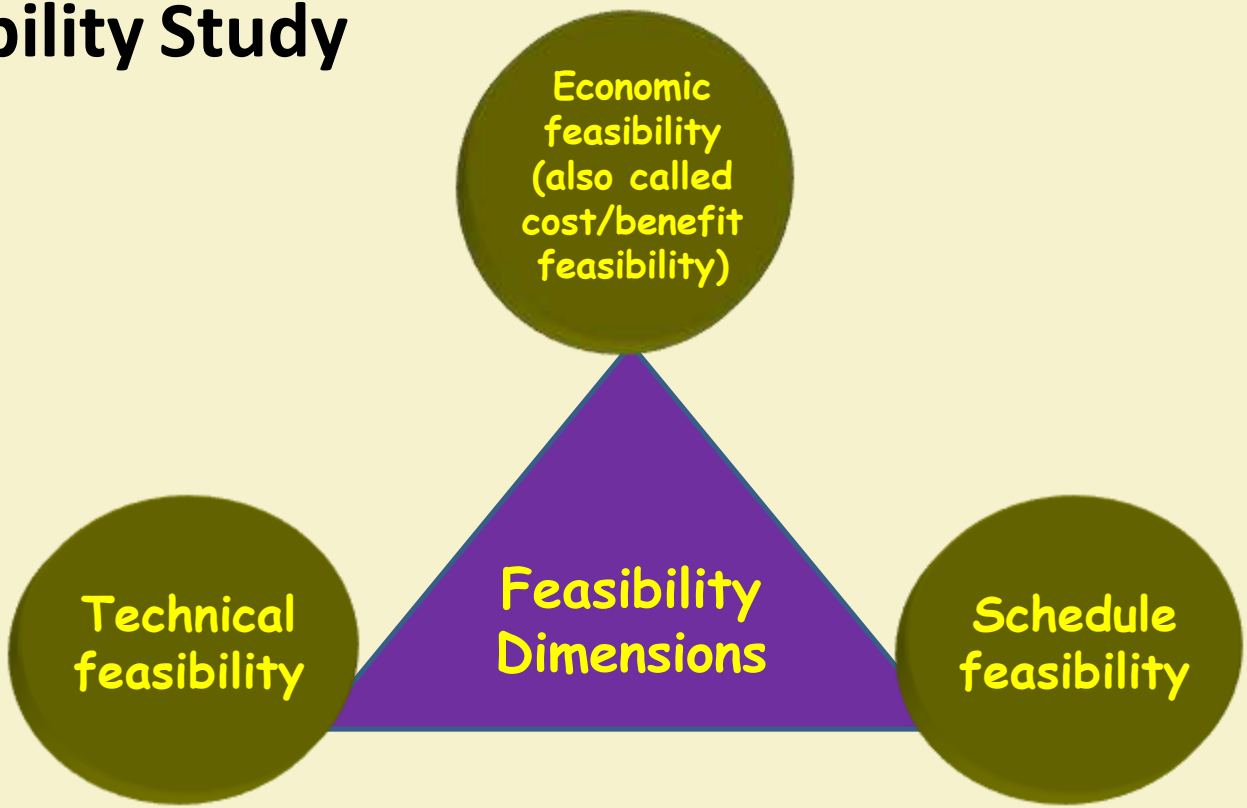
- Most organizations usually define:
  - Standards on the outputs (deliverables) produced at the end of every phase
  - Entry and exit criteria for every phase.
- They also prescribe methodologies for:
  - Specification,
  - Design,
  - Testing,
  - Project management, etc.

Process Model

Classical Waterfall Model (CONT.)

- The guidelines and methodologies of an organization:
  - –Called the organization's software development methodology.
- Software development organizations:
  - – Expect fresh engineers to master the organization's software development methodology.

Feasibility Study



- Main aim of feasibility study: determine whether developing the software is:
  - **Financially worthwhile**
  - **Technically feasible.**

Feasibility Study

- 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.**

First Step

- SPF Scheme for CFL
- CFL has a large number of employees, exceeding 50,000.
- Majority of these are casual labourers
- Mining being a risky profession:
  - Casualties are high
- Though there is a PF:
  - But settlement time is high
- There is a need of SPF:
  - For faster disbursement of benefits

Case Study

Feasibility: Case Study

- Manager visits main office, finds out the main functionalities required
- Visits mine site, finds out the data to be input
- Suggests alternate solutions
- Determines the best solution
- Presents to the CFL Officials
- Go/No-Go Decision

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.

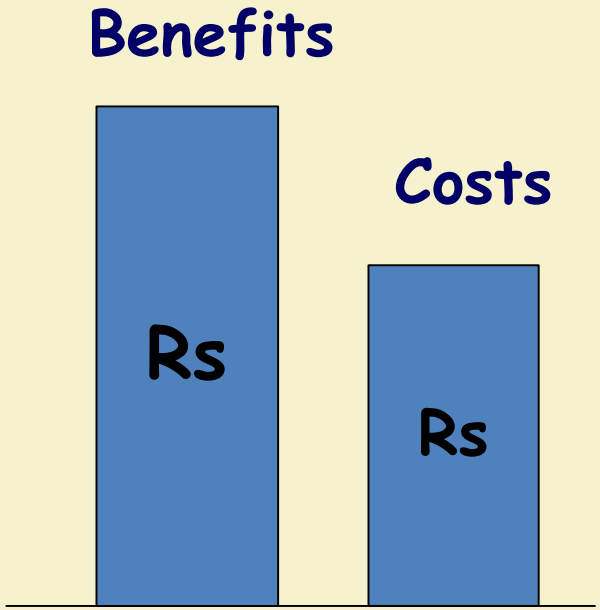
- Perform a cost/benefit analysis:
  - –Determine which solution is the best.
  - –May also find that none of the solutions is feasible due to:
    - high cost,
    - resource constraints,
    - technical reasons.

Activities during Feasibility Study

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

The business case



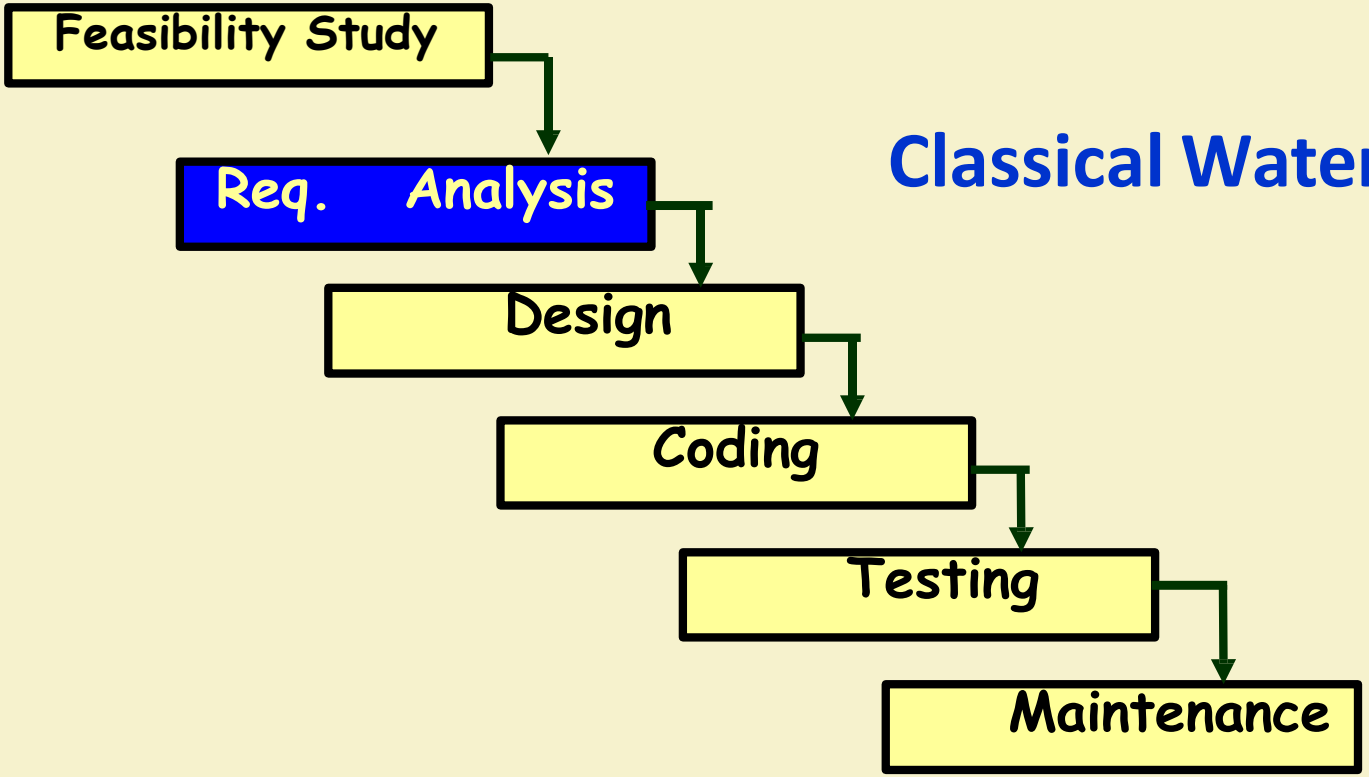
- Benefits of delivered project must outweigh costs
- Costs include:
  - Development
  - Operation
- Benefits:
  - Quantifiable
  - Non-quantifiable

The business case

- 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

Writing an Effective Business Case

1. **Executive summary**
2. **Project background:**
   - The focus must be on what, exactly, the project is undertaking, and should not be confused with what might be a bigger picture.
3. **Business opportunity**
   - What difference will it make?
   - What if we don't do it?
4. **Costs**
   - Should include the cost of development, implementation, training, change management, and operations.
5. **Benefits**
   - Benefits usually presented in terms of revenue generation and cost reductions.
6. **Risks**
   - Identify risks.
   - Explain how these will be managed.



Classical Waterfall Model

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.

Requirements Analysis and Specification

- Gather requirements data from the customer:
  - –Analyze the collected data to understand what customer wants
- Remove requirements problems:
  - –Inconsistencies
  - –Anomalies
  - –Incompleteness
- Organize into a Software Requirements Specification (SRS) document.

Requirements Gathering

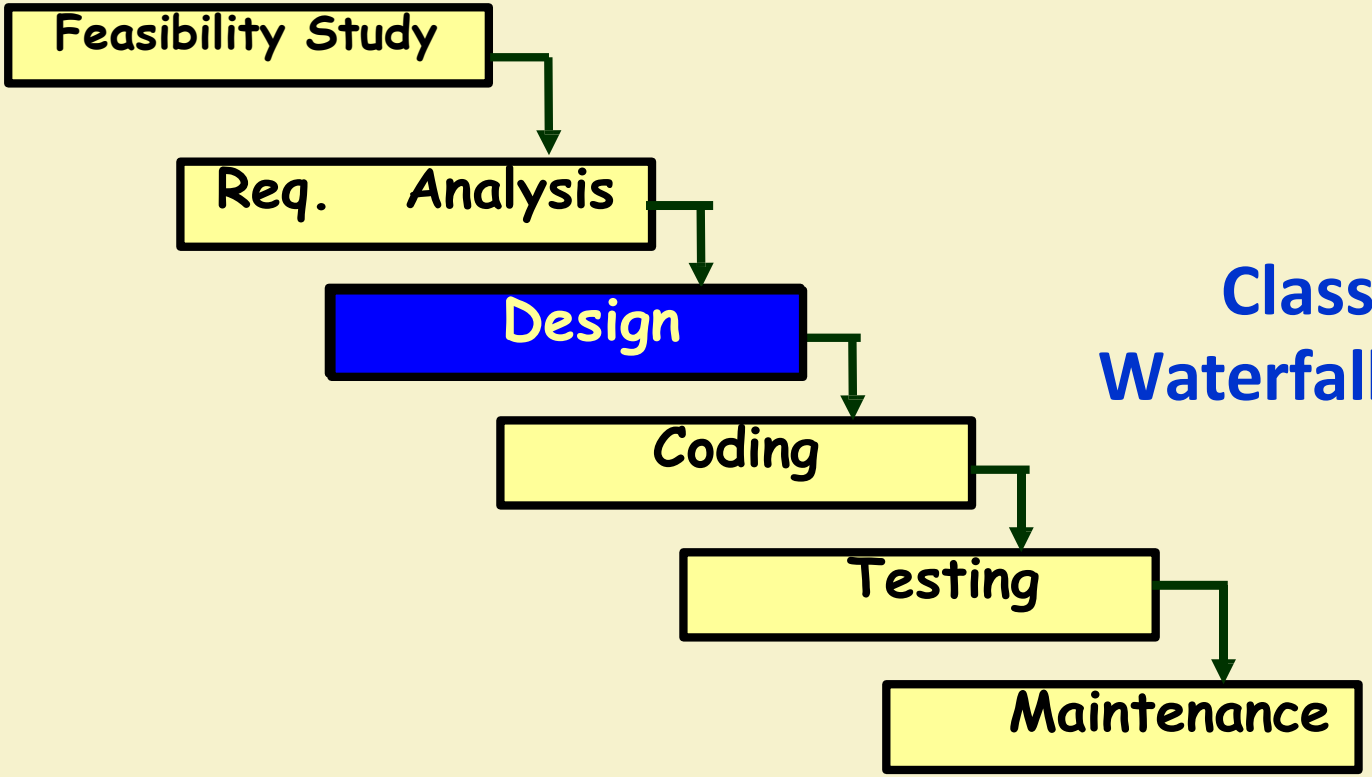
- Gathering relevant data:
  - Usually collected from the end-users through interviews and discussions.
  - Example: for a business accounting software:
    - Interview all the accountants of the organization to find out their requirements.

Requirements Analysis (Cont...)

- The data you initially collect from the users:
  - Usually contain several contradictions and ambiguities.
  - Why?
  - **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.



Classical Waterfall Model

Design

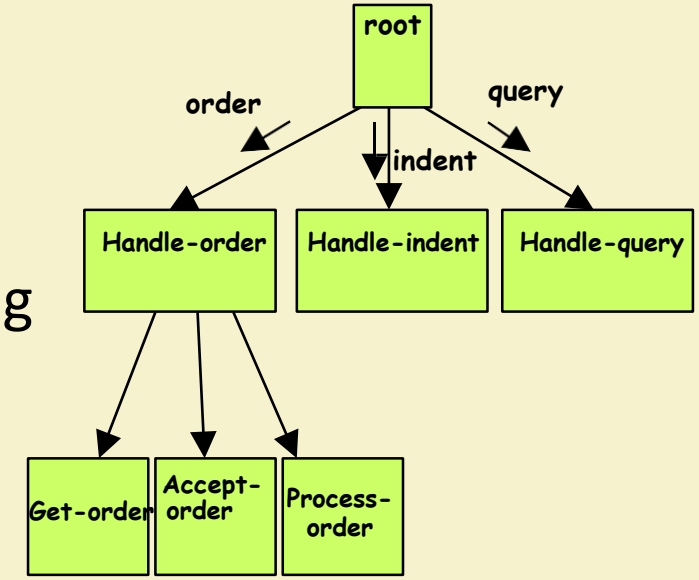
- During design phase requirements specification is transformed into :
  - A form suitable for implementation in some programming language.
- Two commonly used design approaches:
  - Traditional approach,
  - Object oriented approach

Traditional Design Approach

- Consists of two activities:
  - Structured analysis (typically carried out by using DFD)
  - Structured design

Structured Design

- High-level design:
  - decompose the system into modules,
  - represent invocation relationships among the modules.
- Detailed design:
  - different modules designed in greater detail:
    - data structures and algorithms for each module are designed.

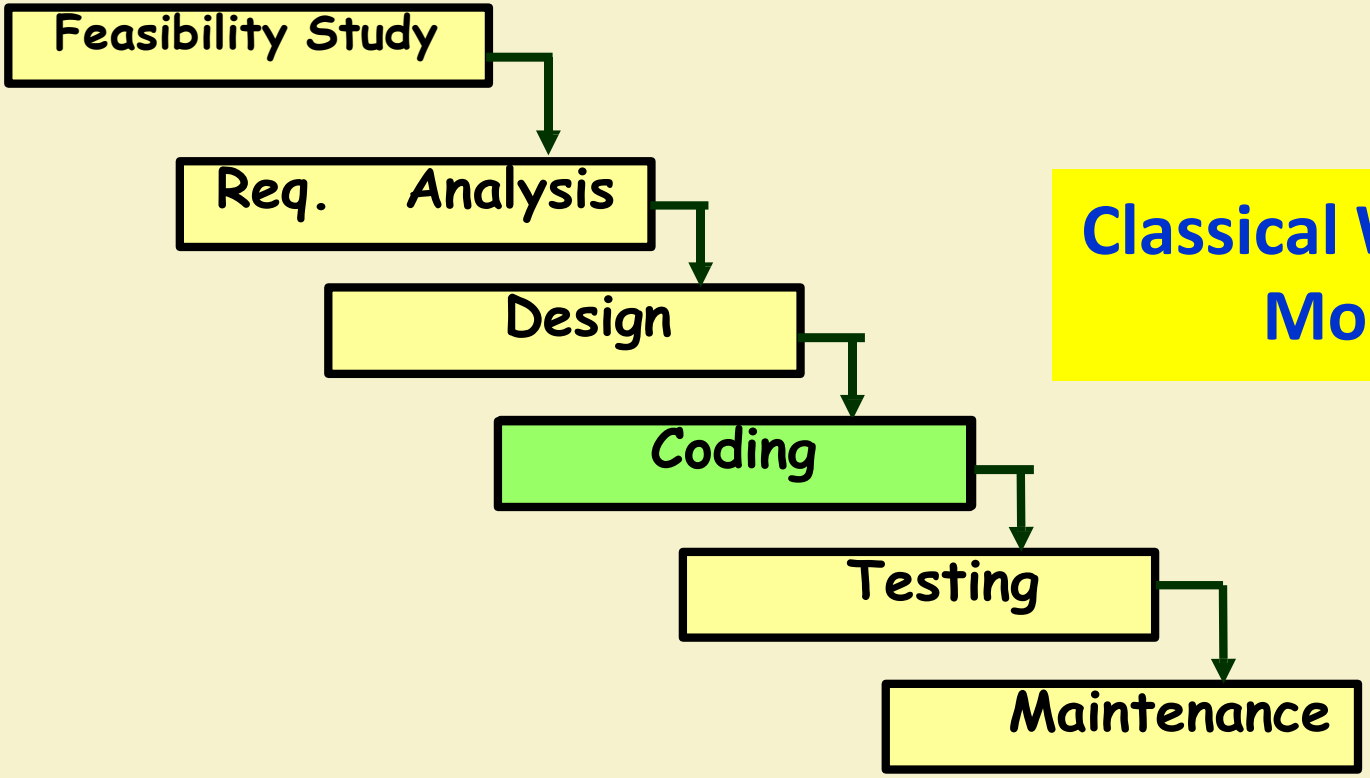


- First identify various objects (real world entities) occurring in the problem:
  - Identify the relationships among the objects.
  - For example, the objects in a pay-roll software may be:
    - employees,
    - managers,
    - pay-roll register,
    - Departments, etc.

Object-Oriented Design

Object Oriented Design (CONT.)

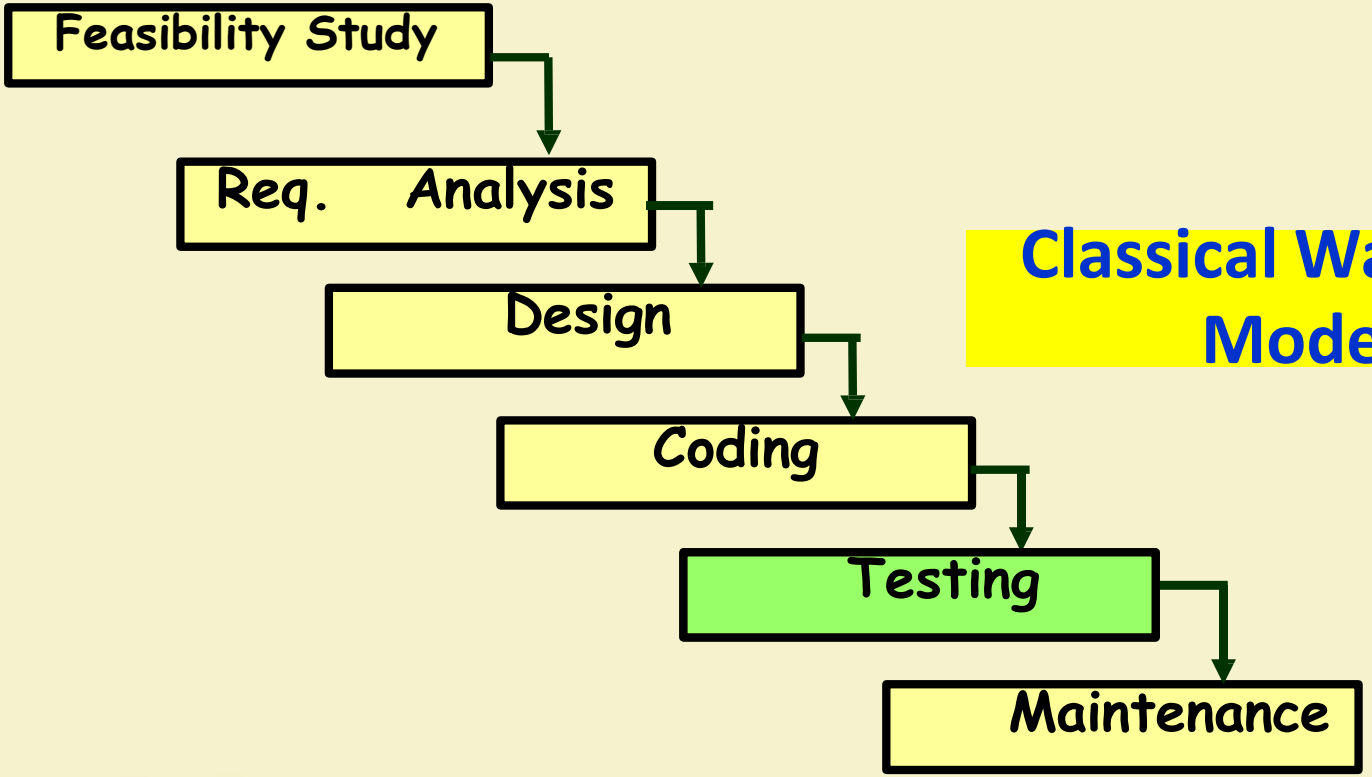
- Object structure:
  - Refined to obtain the detailed design.
- OOD has several advantages:
  - Lower development effort,
  - Lower development time,
  - Better maintainability.



Classical Waterfall Model

Coding and Unit Testing

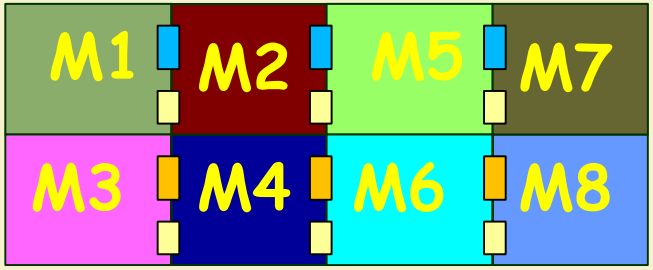
- During this phase:
  - –Each module of the design is coded,
  - –Each module is unit tested
    - That is, tested independently as a stand alone unit, and debugged.
  - –Each module is documented.



Classical Waterfall Model

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.



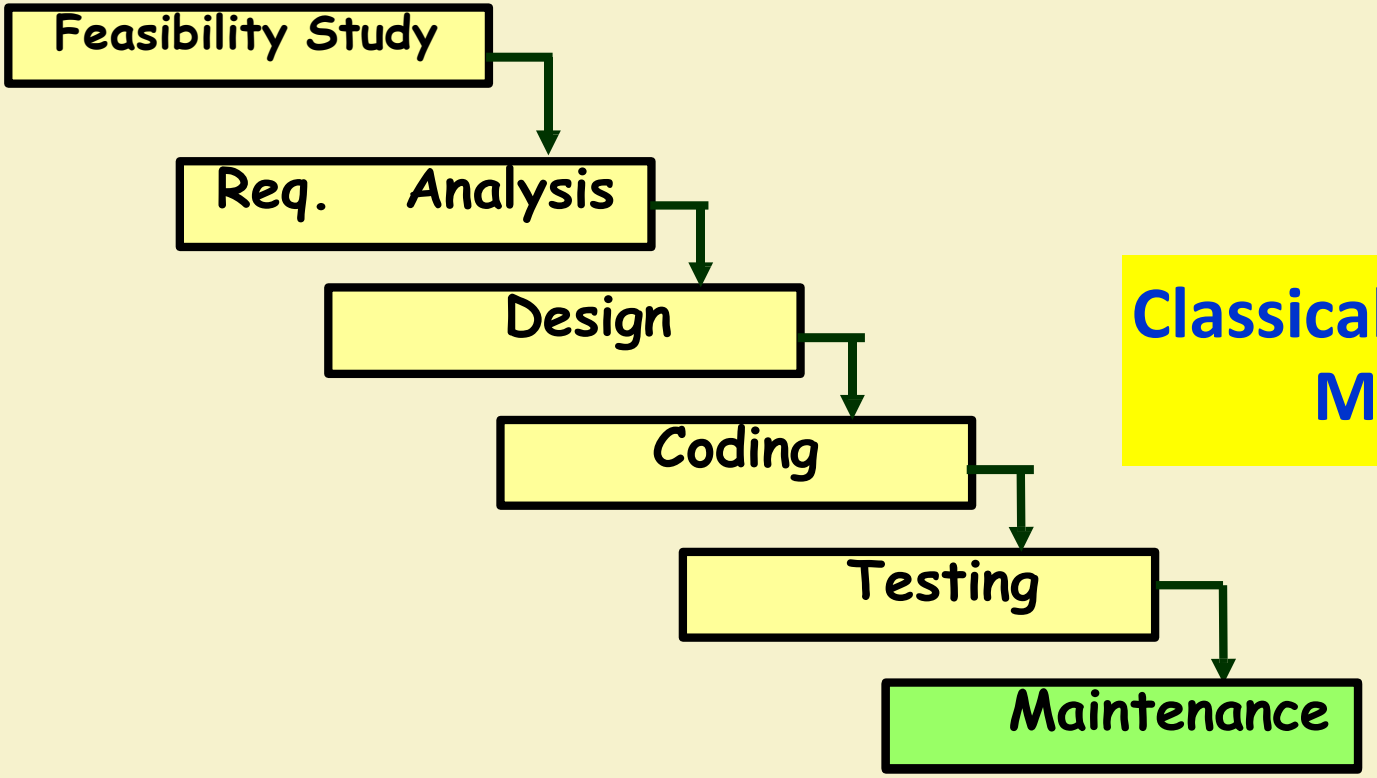
System Testing

- 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.



Classical Waterfall Model

Maintenance

- 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
  - 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.

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.

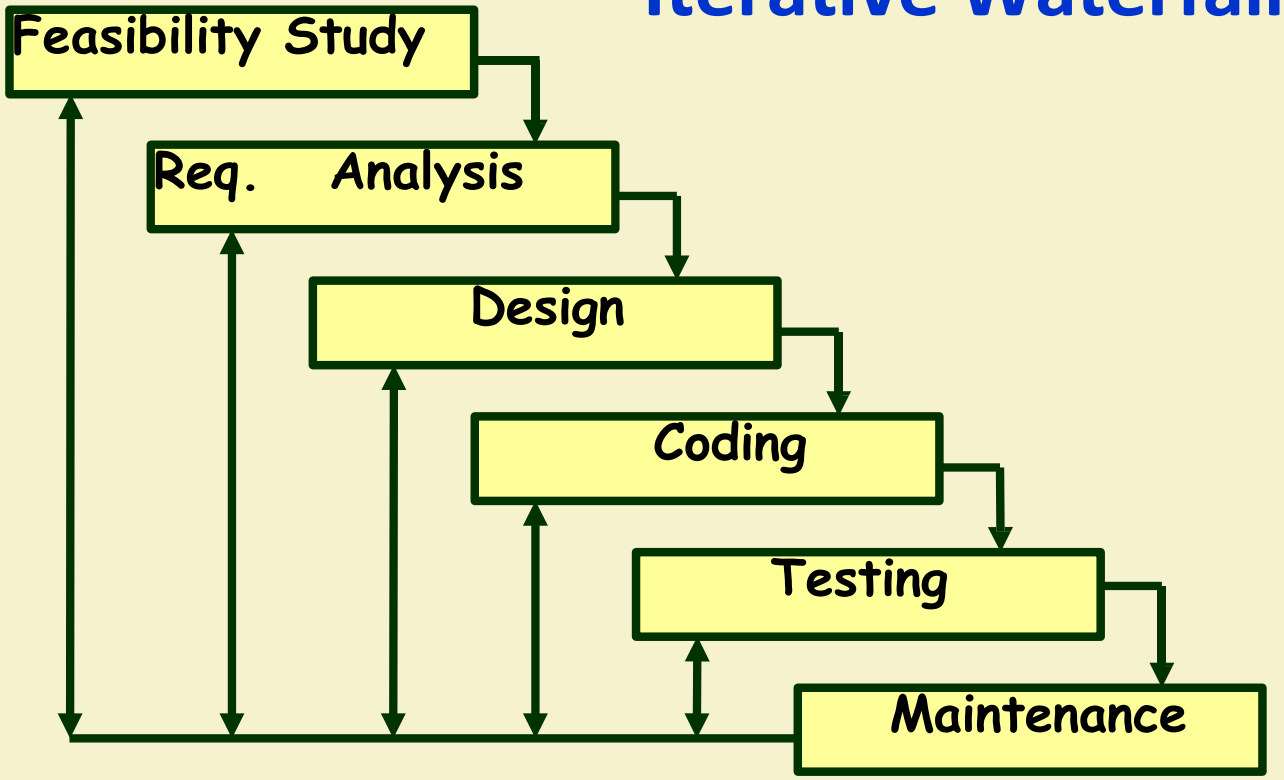
Iterative Waterfall Model (CONT.)

- Defects usually get detected much later in the life cycle:
  - For example, a design defect might go unnoticed till the coding or testing phase.
  - **The later the phase in which the defect gets detected, the more expensive is its removal --- why?**

Iterative Waterfall Model (CONT.)

- Once a defect is detected:
  - The phase in which it occurred needs to be reworked.
  - Redo some of the work done during that and all subsequent phases.
- Therefore need feedback paths in the classical waterfall model.

Iterative Waterfall Model (CONT.)



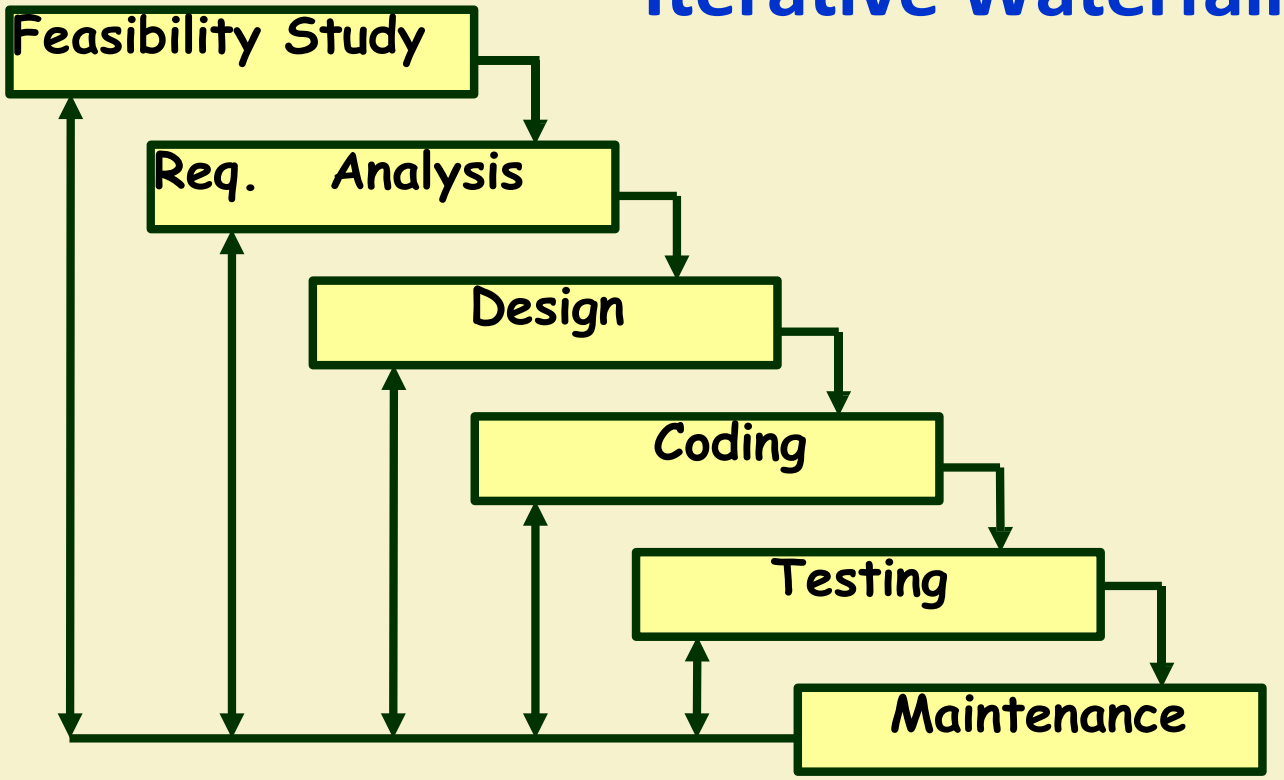
Phase Containment of Errors (Cont...)

- **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.

Iterative Waterfall Model (CONT.)



Waterfall Strengths

- Easy to understand, easy to use, especially by inexperienced staff
- Milestones are well understood by the team
- Provides requirements stability during development
- Facilitates strong management control (plan, staff, track)

Waterfall Deficiencies

- All requirements must be known upfront – in **most projects requirement change occurs after project start**
- Can give a false impression of progress
- Integration is one big bang at the end
- Little opportunity for customer to pre-view the system.

When to use the Waterfall Model?

- Requirements are well known and stable
- Technology is understood
- Development team have experience with similar projects

Classical Waterfall Model (CONT.)

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

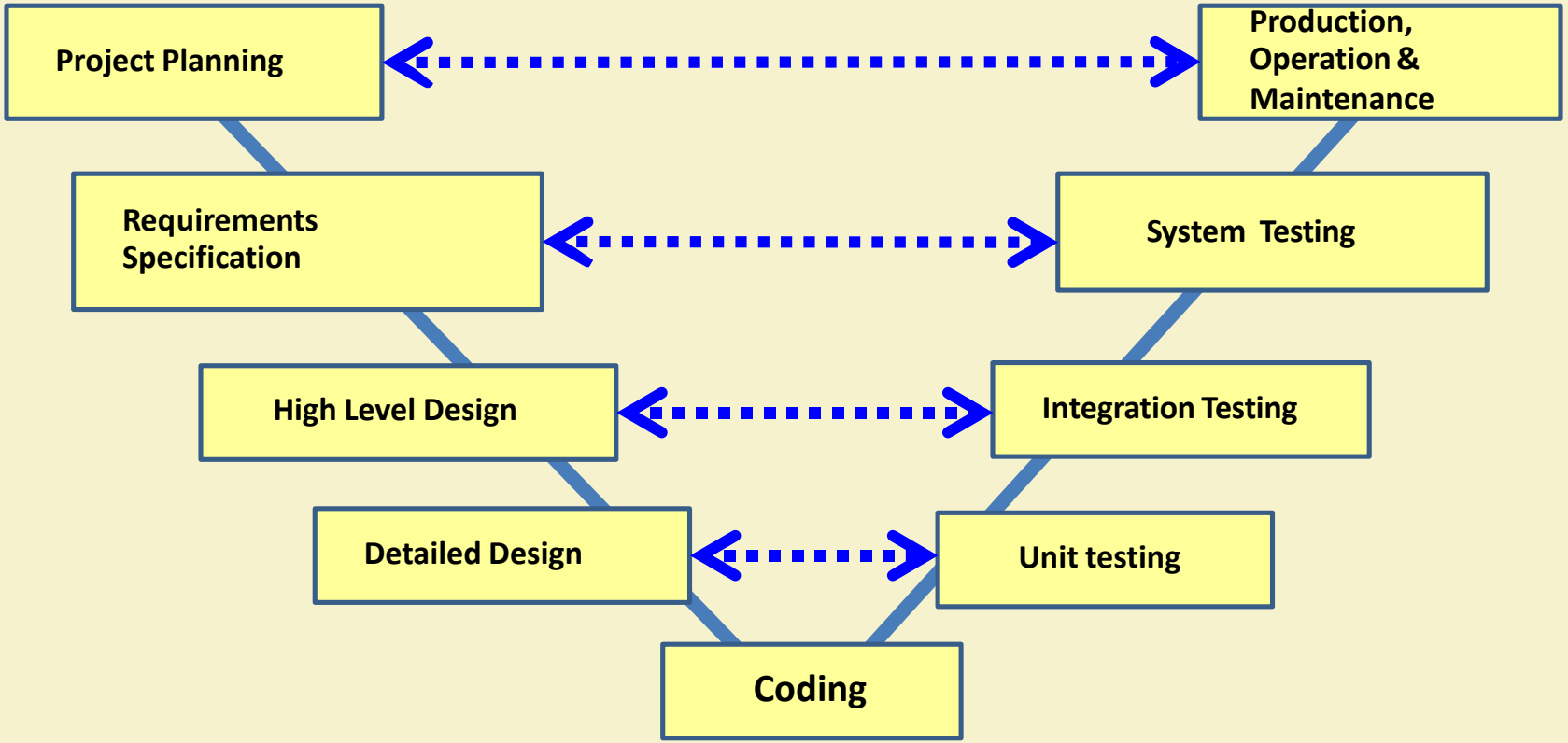
Classical Waterfall Model (CONT.)

- Metaphor of mathematical theorem proving:
  - –A mathematician presents a proof as a single chain of deductions,
    - Even though the proof might have come from a convoluted set of partial attempts, blind alleys and backtracks.

V Model

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.


Project Planning
Production, Operation & Maintenance
Requirements Specification
System Testing
High Level Design
Integration Testing
Detailed Design
Unit testing
Coding

V Model Steps

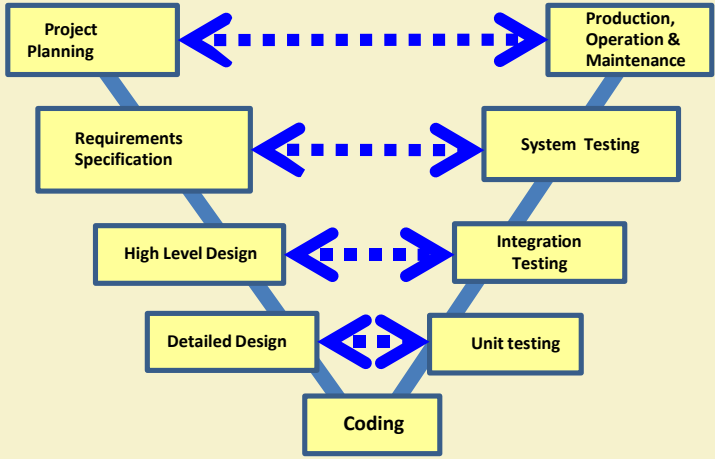
- Planning
- Requirements Analysis and Specification
  - System test design
- High-level Design
  - Integration Test dsign
- Detailed Design
  - Unit test design

V Model: Strengths

- Starting from early stages of software development:
  - **Emphasizes planning for verification and validation of the software**
- Each deliverable is made testable
- Easy to use

V Model Weaknesses

- Does not support overlapping of phases
- Does not handle iterations or phases
- Does not easily accommodate later changes to requirements



- Does not provide support for effective risk handling

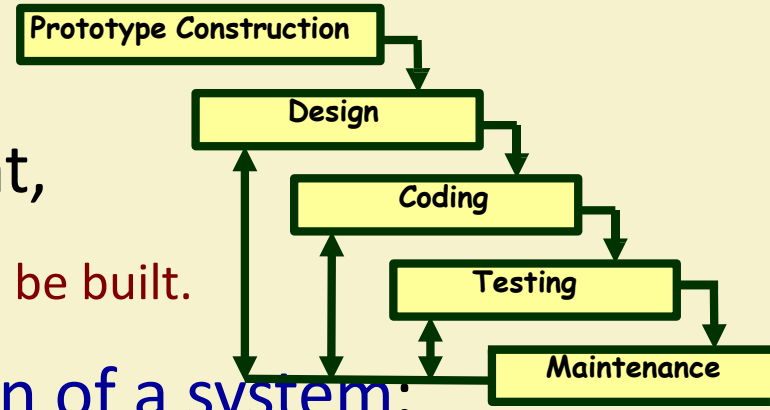
When to use V Model

- Natural choice for systems requiring high reliability:
  - Embedded control applications, safety-critical software
- All requirements are known up-front
- Solution and technology are known

Prototyping Model

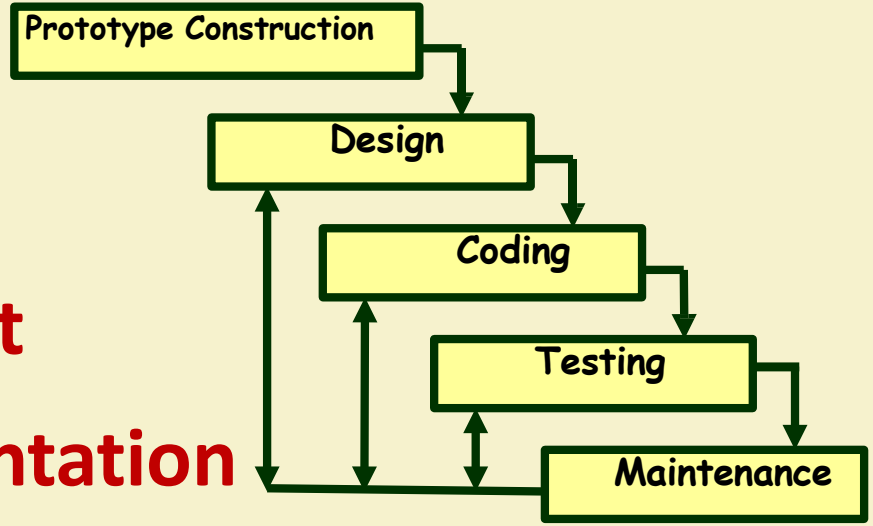
Prototyping Model

- 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**



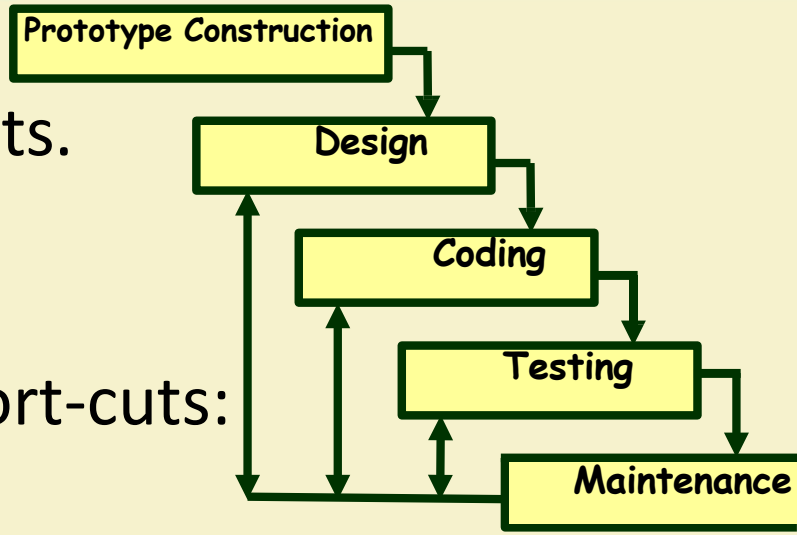
Reasons for Developing a Prototype

- **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.)

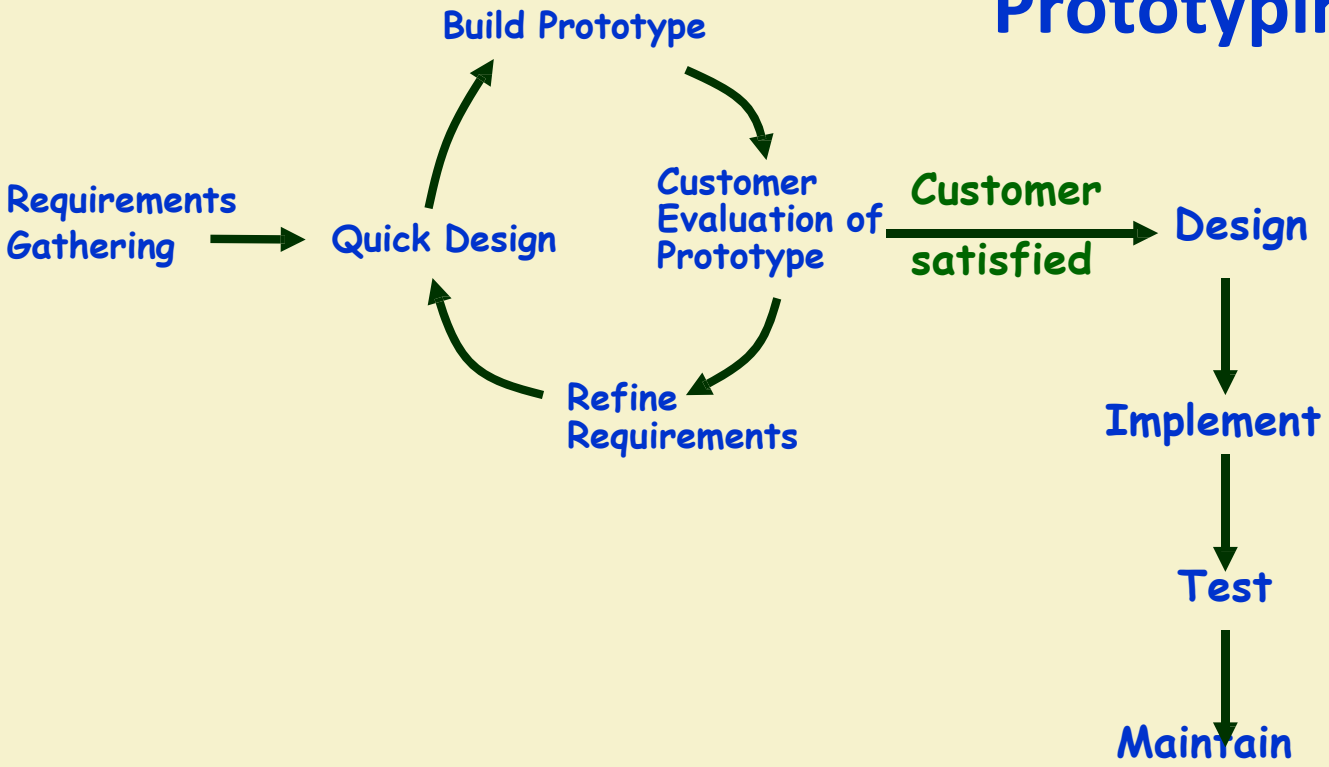
- Another reason for developing a prototype:
  - **It is impossible to "get it right" the first time,**
  - We must plan to throw away the first version:
    - If we want to develop a good software.

Prototyping Model



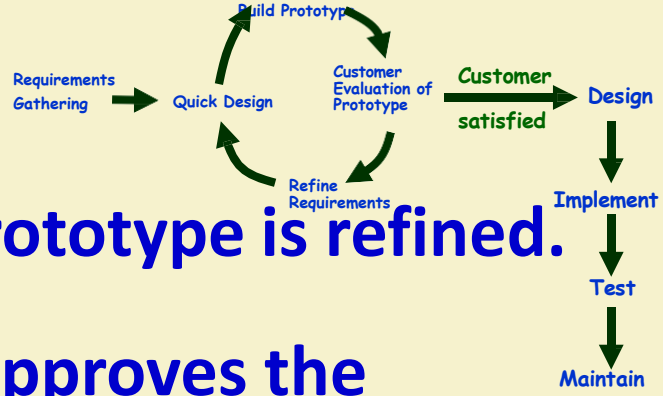
- Start with approximate requirements.
- Carry out a quick design.
- Prototype is built using several short-cuts:
  - Short-cuts might involve:
    - Using inefficient, inaccurate, or dummy functions.
    - A table look-up rather than performing the actual computations.

Prototyping Model (CONT.)



Prototyping Model (CONT.)

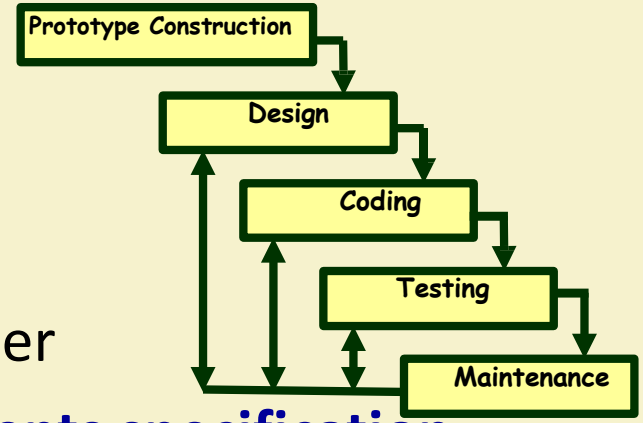
- The developed prototype is submitted to the customer for his evaluation:



  - **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.

Prototyping Model



- Requirements analysis and specification phase becomes redundant:
  - Final working prototype (incorporating all user feedbacks) serves as an **animated requirements specification.**
- **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.

Prototyping Model (CONT.)

- 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.

Prototyping: advantages

- The resulting software is usually more usable
- User needs are better accommodated
- The design is of higher quality
- The resulting software is easier to maintain
- Overall, the development incurs less cost

Prototyping: disadvantages

- For some projects, it is expensive
- Susceptible to over-engineering:
  - Designers start to incorporate sophistications that they could not incorporate in the prototype.

Major difficulties of Waterfall-Based Models

1. Difficulty in accommodating change requests during development.
   - **40% of the requirements change during development**
2. High cost incurred in developing custom applications.
3. **"Heavy weight processes."**

Major difficulties of Waterfall-Based Life Cycle Models

- Requirements for the system are determined at the start:
  - Are assumed to be fixed from that point on.
  - Long term planning is made based on this.

"… the assumption that one can specify a satisfactory system in advance, get bids for its construction, have it built, and install it. …this assumption is fundamentally wrong and many software acquisition problems spring from this…" **Frederick Brooks**

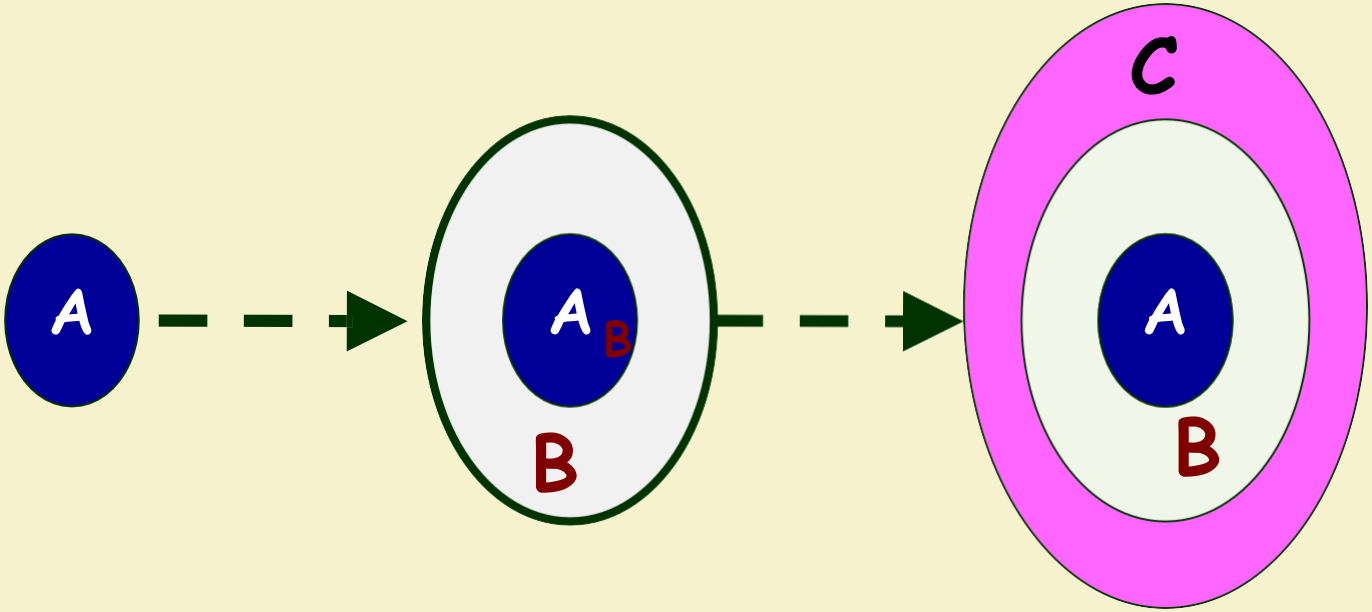
Incremental Model

- “The basic idea… take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system. Learning comes from both the development and use of the system… Start with a simple implementation of a subset of the software requirements and iteratively enhance the evolving sequence of versions. At each version design modifications are made along with adding new functional capabilities. “ Victor Basili

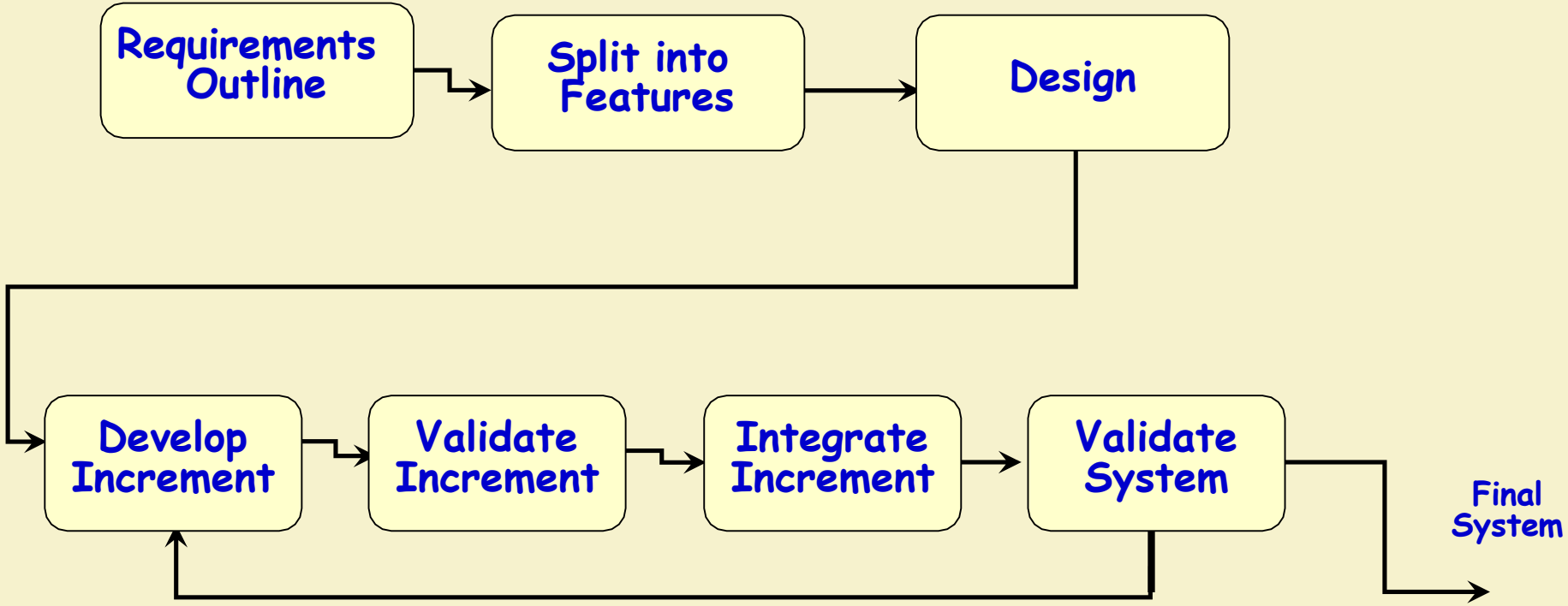
Incremental and Iterative Development (IID)

- **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)

Customer's Perspective

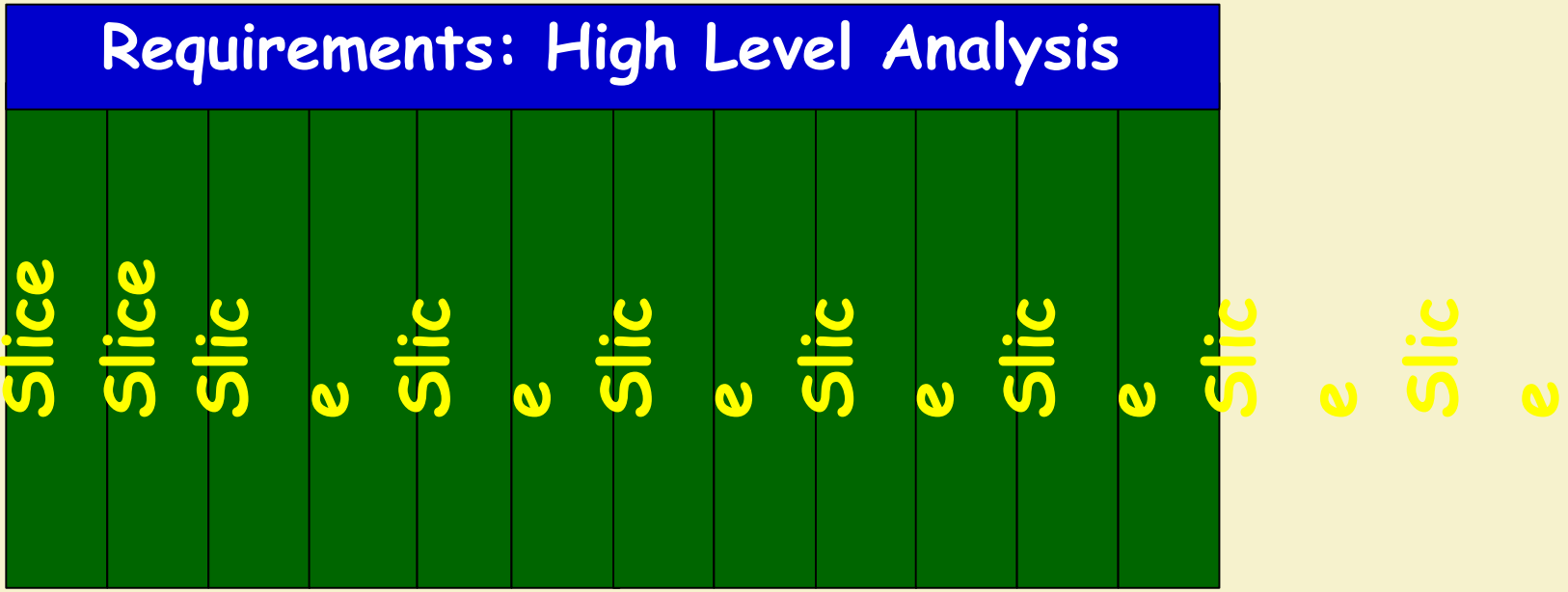


Incremental Model



Incremental Model: Requirements

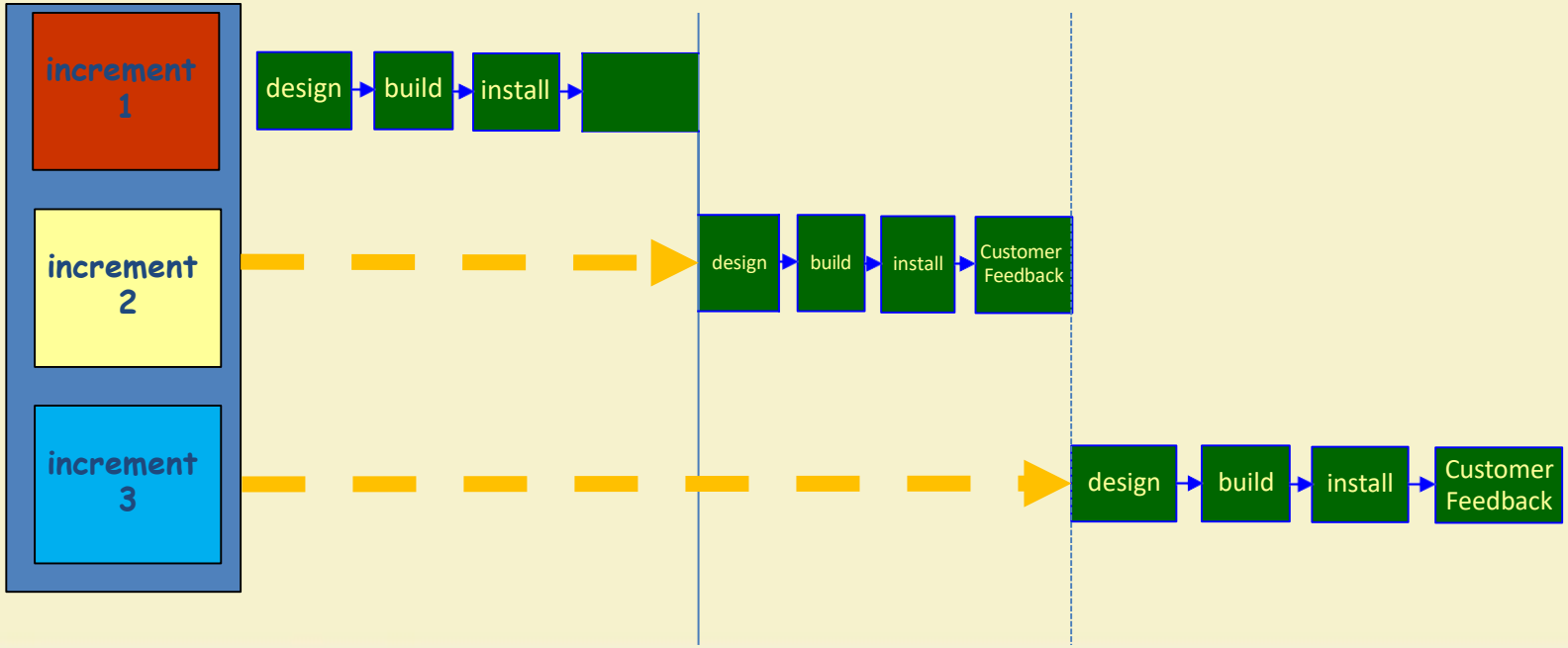
Split into Features



Incremental Model

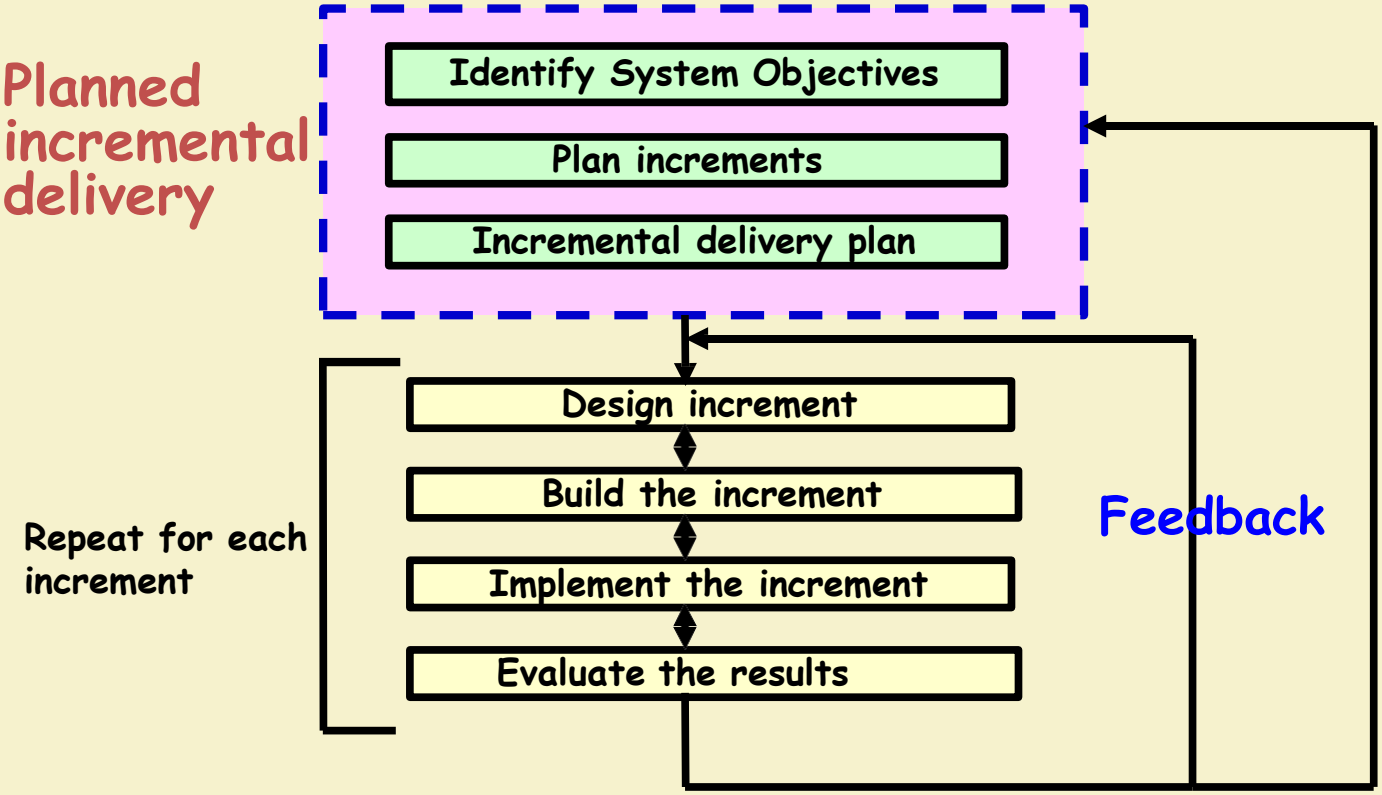
- Waterfall: single release
- Iterative: many releases (increments)
  - **First increment: core functionality**
  - **Successive increments: add/fix functionality**
  - **Final increment: the complete product**
- Each iteration: a short mini-project with a separate lifecycle
  - e.g., waterfall

Incremental delivery



The incremental process

Planned incremental delivery



Which step first?

- Some steps will be pre-requisite because of physical dependencies
- Others may be in any order
- Value to cost ratios may be used
  - V/C where
  - V is a score 1-10 representing value to customer
  - C is a score 0-10 representing cost to developers

V/C ratios: an example

step	value	cost	ratio	
profit reports	9	2	4.5	2nd
online database	1	9	0.11	5th
ad hoc enquiry	5	5	1	4th
purchasing plans	9	4	2.25	3rd
profit-based pay for managers	9	1	9	1st

Evolutionary Model with Iterations

An Evolutionary and Iterative Development Process...

- 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

Evolutionary Model with Iteration

- "A complex system will be most successful if implemented in small steps… "retreat" to a previous successful step on failure… opportunity to receive some feedback from the real world before throwing in all resources… and you can correct possible errors…" Tom Glib in Software Metrics

Evolutionary model with iteration

- Evolutionary iterative development implies that the requirements, plan, estimates, and solution evolve or are refined over the course of the iterations, rather than fully defined and "frozen" in a major up-front specification effort before the development iterations begin. Evolutionary methods are consistent with the pattern of unpredictable discovery and change in new product development." **Craig Larman**

Evolutionary Model

- 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.

Activities in an Iteration

- 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

Evolutionary Model with Iteration

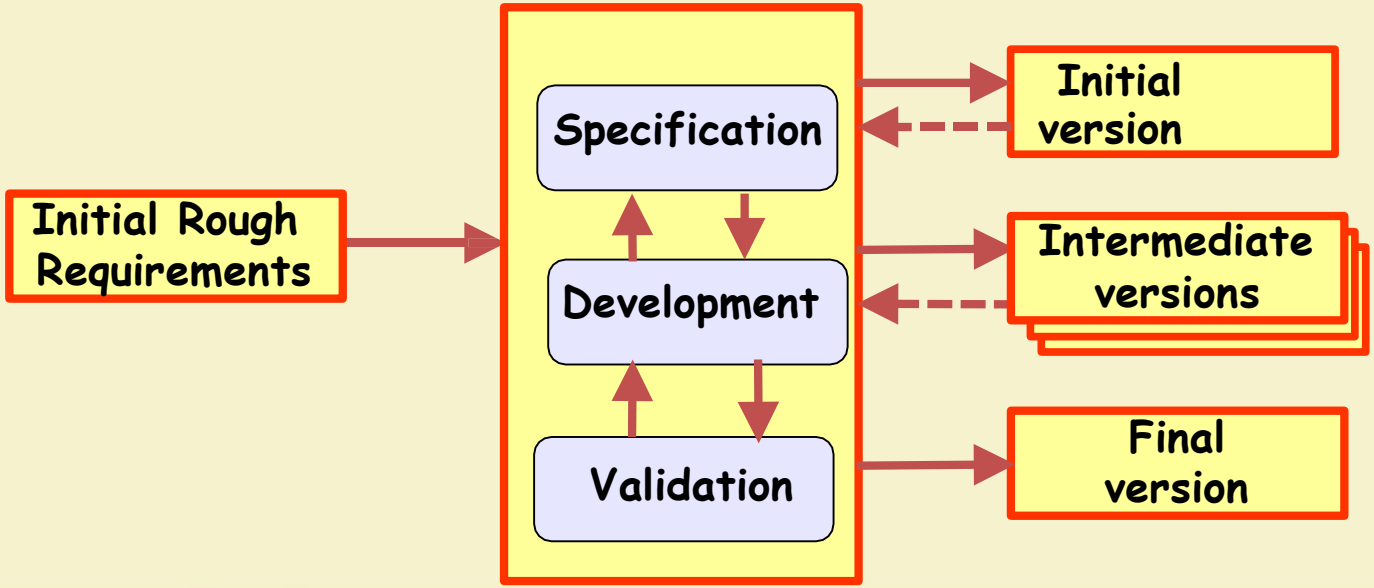
- Outcome of each iteration: tested, integrated, executable system
- Iteration length is short and fixed
  - **Usually between 2 and 6 weeks**
  - **Development takes many iterations (for example: 10-15)**
- Does not "freeze" requirements and then conservatively design :
  - **Opportunity exists to modify requirements as well as the design…**

Evolutionary Model (CONT.)

- 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.

Evolutionary Model

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



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:**
  - Software more likely to meet exact user requirements.
- **Core modules get tested thoroughly:**
  - Reduces chances of errors in final delivered software.

Advantages of evolutionary model

- 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.

Advantages of Evolutionary Model with Iteration

- Training can start on an earlier release

  –customer feedback taken into account

- Frequent releases allow developers to fix unanticipated problems quicker.

Evolutionary Model: Problems

- **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.**

RAD Model

Rapid Application Development (RAD) Model

- 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.

Important Underlying Principle

- 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.**

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.

Methodology

- 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).**

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 RAD is Unsuitable?

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

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.

Prototyping versus RAD

- 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.

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.

RAD versus Iterative Waterfall Model

- 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.

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.

Unified Process

Unified Process

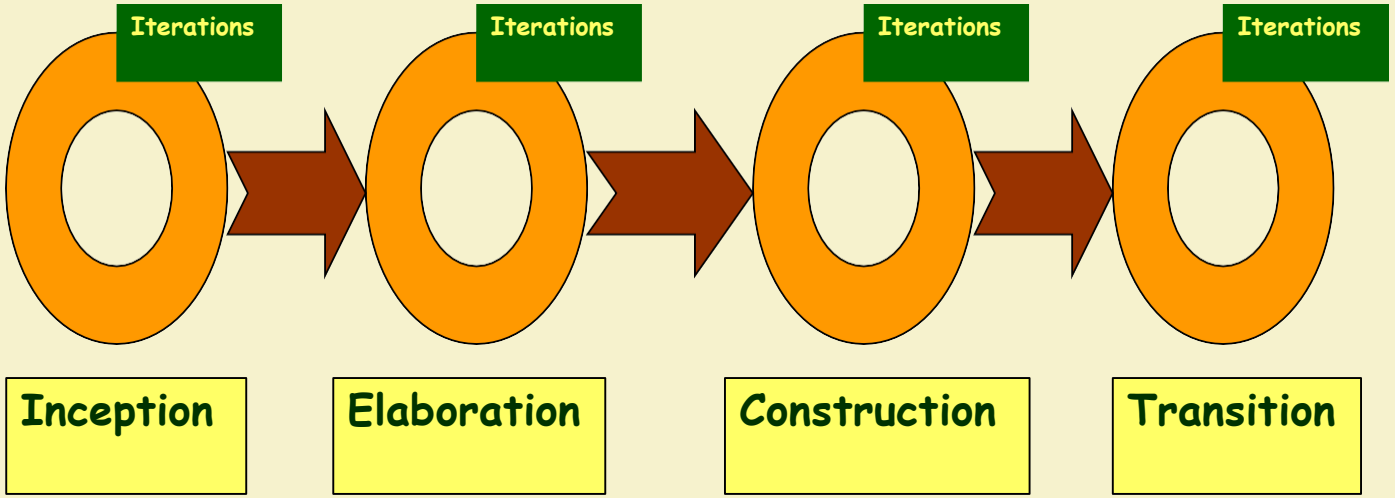
- Developed Ivar Jacobson, Grady Booch and James Rumbaugh
  - Incremental and iterative
- Rational Unified Process (RUP) is version tailored by Rational Software:
  - Acquired by IBM in February 2003.

Four Phases --- and iterative Development at Every phase

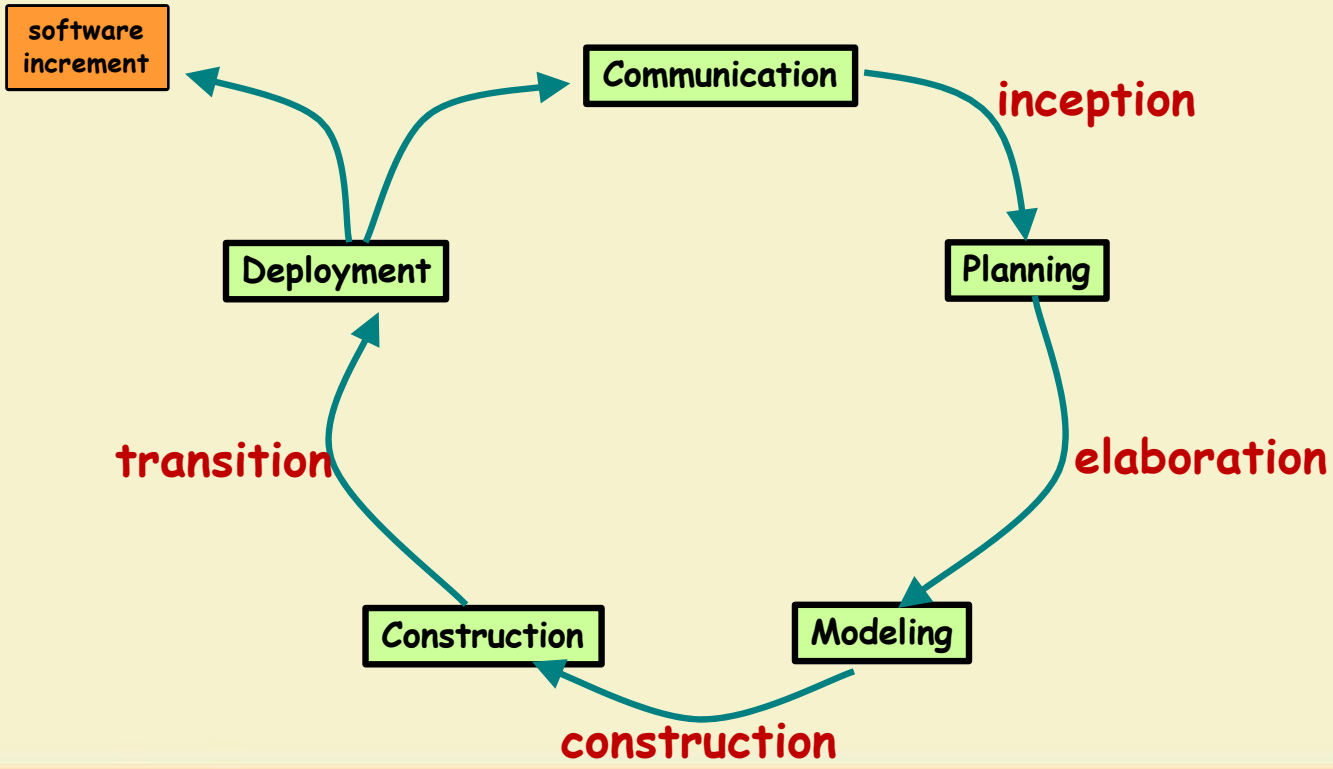
- Inception Phase
- Elaboration Phase
- Construction Phase
- Transition Phase

Unified Process Iterations in Phases

The duration of and iteration may vary from two weeks or less.



Unified process



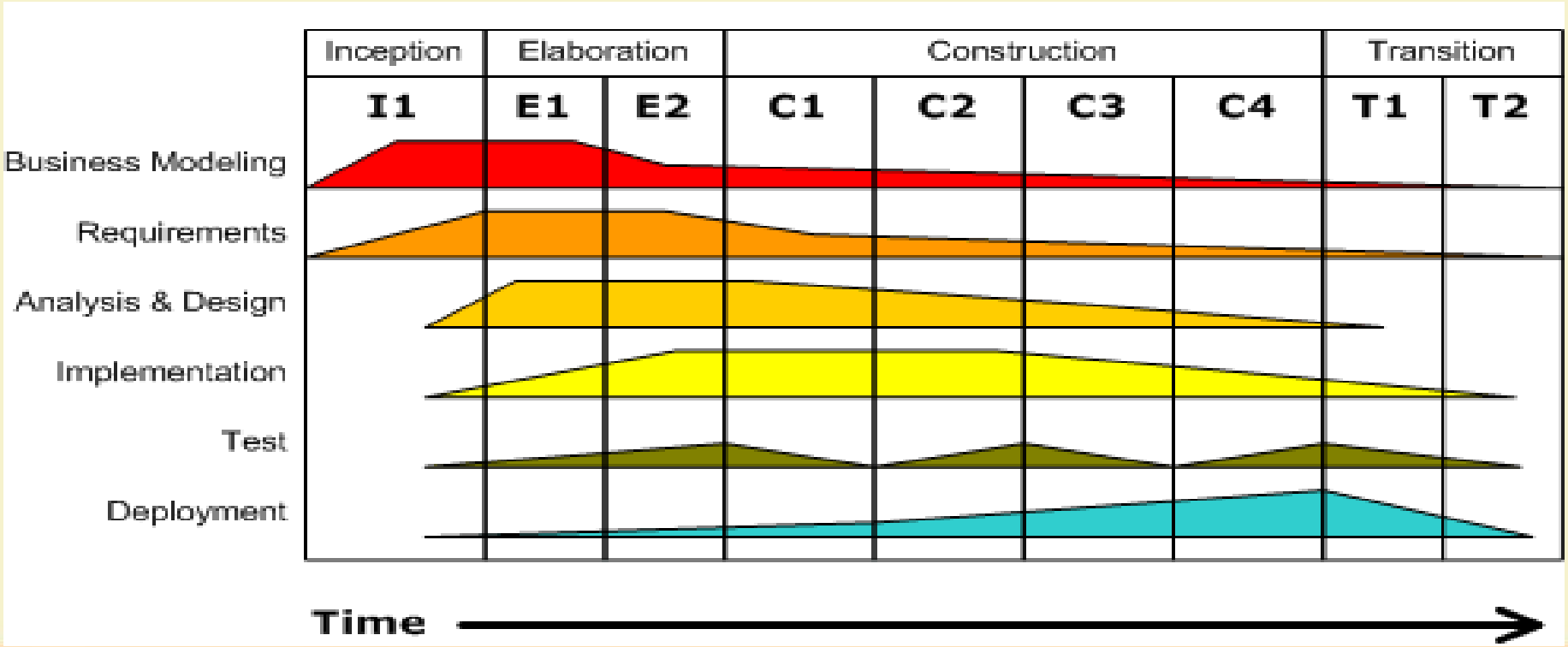
Unified process work products

Inception phase	Elaboration phase	Construction phase	Transition phase
vision document initial use-case model initial business case initial risk list project plan prototype(s) ...	use-case model requirements analysis model preliminary model revised risk list preliminary manual ...	design model SW components test plan test procedure test cases user manual installation manual ...	SW increment beta test reports user feedback ...

Structure of RUP Process

- Two dimensions.
- Horizontal axis:
  - Represents time and shows the lifecycle aspects of the process.
- Vertical axis:
  - Represents core process workflows.

Two dimensions of Unified Process



Inception activities

- Formulate scope of project
- Risk management, staffing, project plan
- Synthesize a candidate architecture.

- Initial requirements capture
- Cost Benefit Analysis
- Initial Risk Analysis
- Project scope definition
- Defining a candidate architecture
- Development of a disposable prototype
- Initial Use Case Model (10% - 20% complete)
- First pass Domain Model

Outcome of Inception Phase

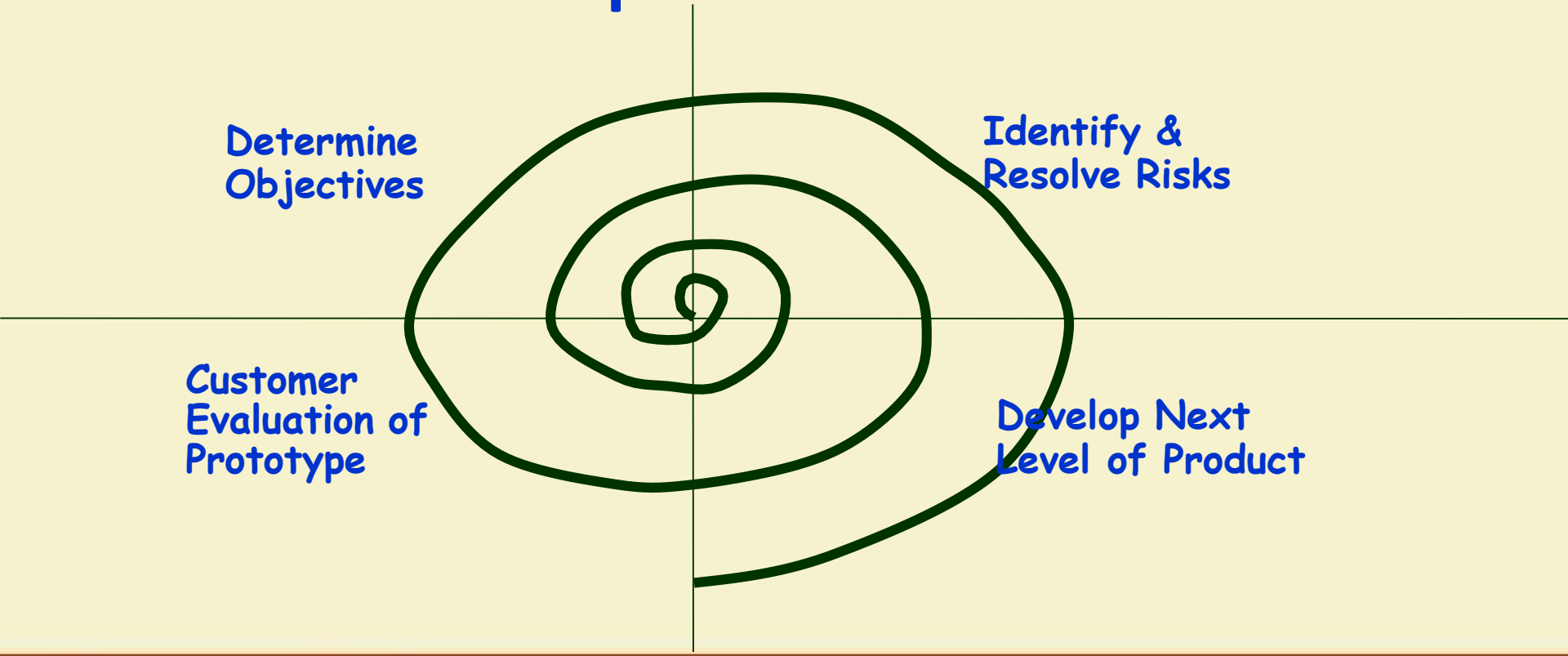
Spiral Model

- 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



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.

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.

- Subsumes all discussed models:
  - a single loop spiral represents waterfall model.
  - uses an evolutionary approach --
    - iterations over 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.

Spiral Model as a Meta Model