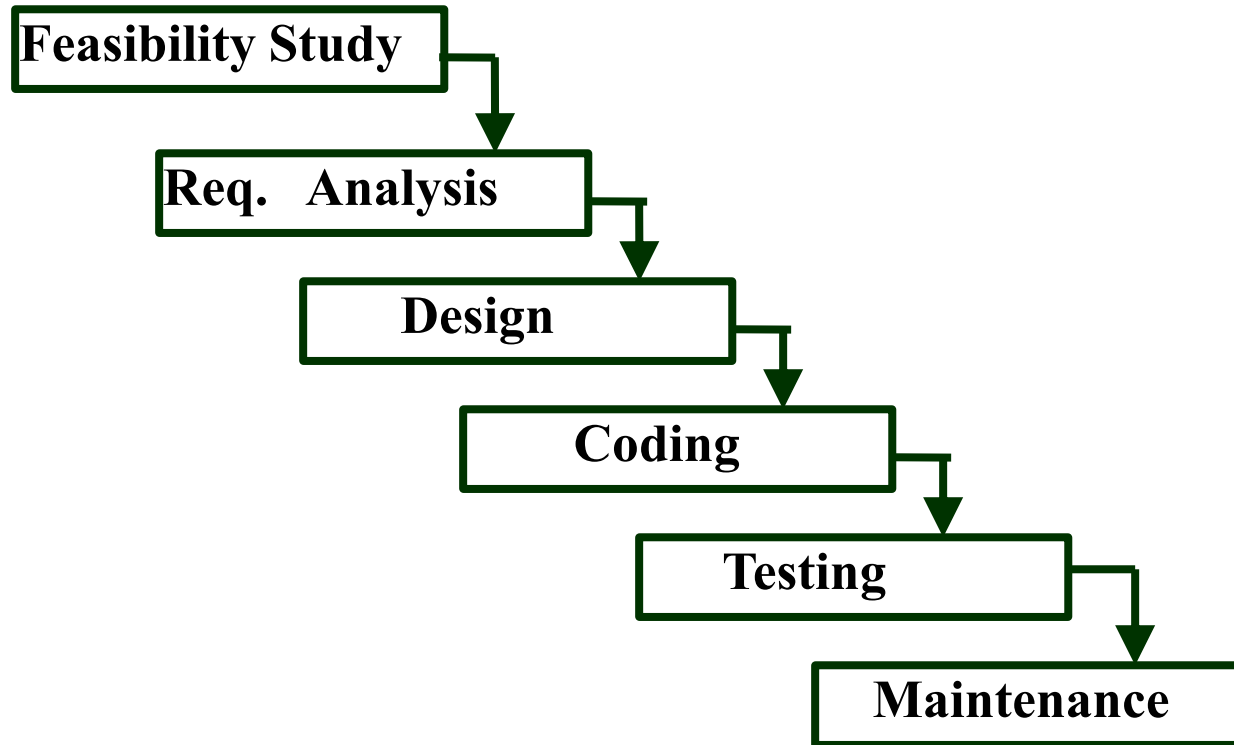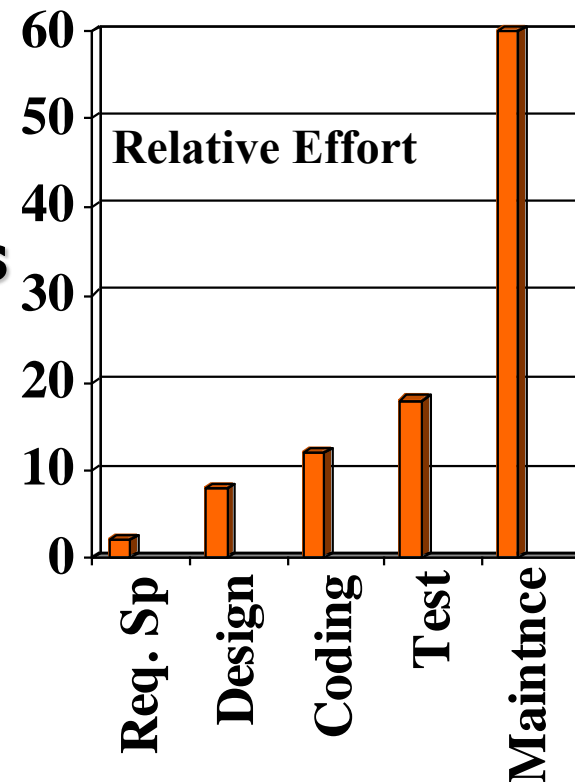# Classical Waterfall Model

- **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

```
┌─────────────────────┐
│  Feasibility Study  │
└─────────────────────┘
            │
            ▼
    ┌─────────────────────┐
    │  Req.   Analysis    │
    └─────────────────────┘
                │
                ▼
        ┌─────────────────────┐
        │       Design        │
        └─────────────────────┘
                    │
                    ▼
            ┌─────────────────────┐
            │       Coding        │
            └─────────────────────┘
                        │
                        ▼
                ┌─────────────────────┐
                │      Testing        │
                └─────────────────────┘
                            │
                            ▼
                    ┌─────────────────────┐
                    │     Maintenance     │
                    └─────────────────────┘
```

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.

Relative Effort

Chart: Relative Effort by phase — Req. Sp, Design, Coding, Test, Maintnce (y-axis 0–60)

Classical Waterfall Model
(CONT.)

- **Most organizations usually define:**
  - entry and exit criteria for every phase.

- **They also prescribe specific methodologies for:**
  - specification,
  - design,
  - testing,
  - project management, etc.

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.**

Requirements Analysis and Specification

- **<u>Aim of this phase:</u>**
  - **understand the <u>exact requirements</u> 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,**
  - **object oriented approach.**

Traditional Design Approach

- Consists of two activities:
  - Structured analysis
  - Structured design

Structured Analysis Activity

- **Identify all the functions to be performed.**

- **Identify data flow among the functions.**

- **Decompose each function recursively into sub-functions.**
  - **Identify data flow among the subfunctions as well.**

Structured Analysis (CONT.)

- **Carried out using Data flow diagrams (DFDs).**

- **After structured analysis, carry out structured design:**
  - **architectural design (or high-level design)**
  - **detailed design (or low-level design).**

Structured Design

- **<u>High-level design:</u>**
  - **decompose the system into <u>*modules*</u>,**
  - **represent relationships among the modules.**

- **<u>Detailed design:</u>**
  - **different modules designed in greater detail:**
    - ∗ **data structures and algorithms for each module are designed.**

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 (coding 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.**

- **<u>Goal of system testing:</u>**

  – 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.)

- **<u>Corrective maintenance:</u>**
  - Correct errors which were not discovered during the product development phases.

- **<u>Perfective maintenance:</u>**
  - Improve implementation of the system
  - enhance functionalities of the system.

- **<u>Adaptive maintenance:</u>**
  - 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

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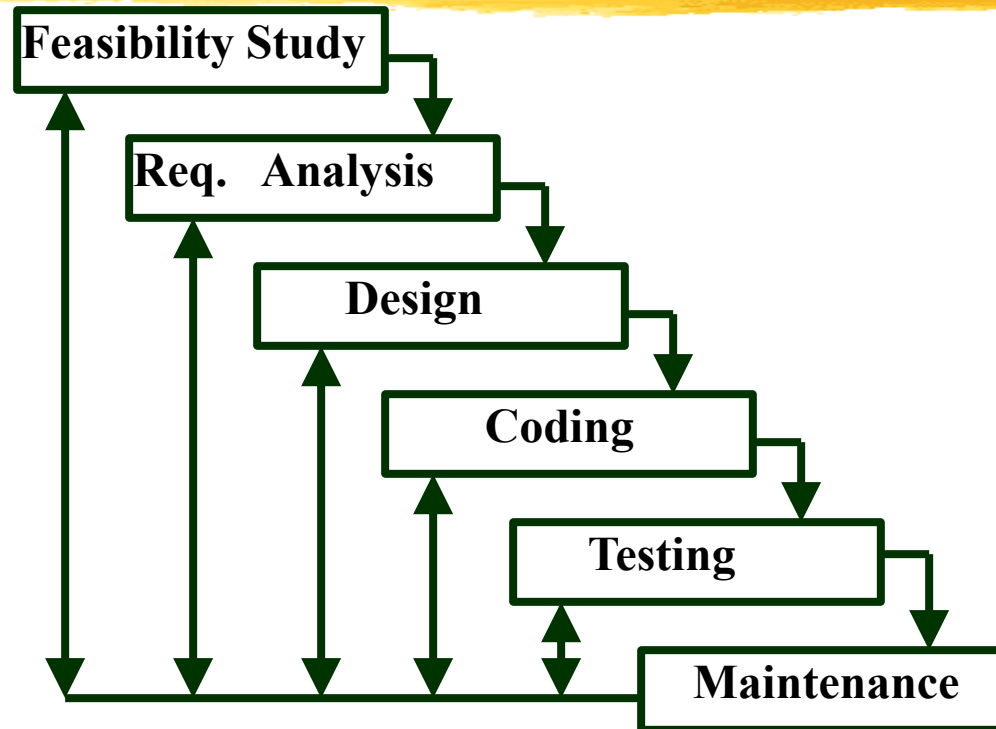
Iterative Waterfall Model
(CONT.)

- **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

Iterative Waterfall Model

- **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, if it is identified at the end of the integration and system testing phase.

34

Phase containment of errors

- **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 most widely used model.**
  - Almost every other model is derived from the waterfall model.

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,
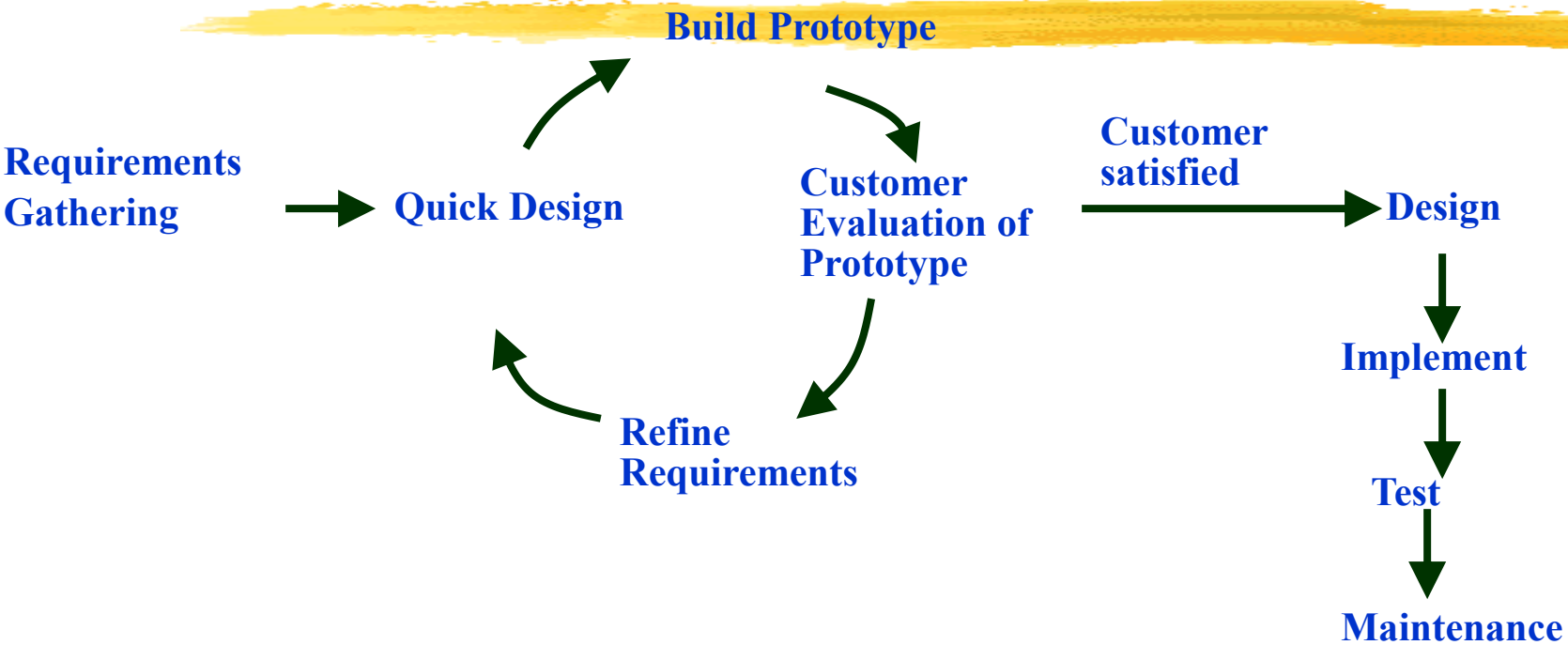 - inefficient performance.

Prototyping Model (CONT.)

- **The 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.)

- **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.)

Build Prototype

Requirements Gathering → **Quick Design** → **Build Prototype** → **Customer Evaluation of Prototype** → **Refine Requirements** → **Quick Design**

Customer satisfied → **Design**

Design → **Implement** → **Test** → **Maintenance**

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.
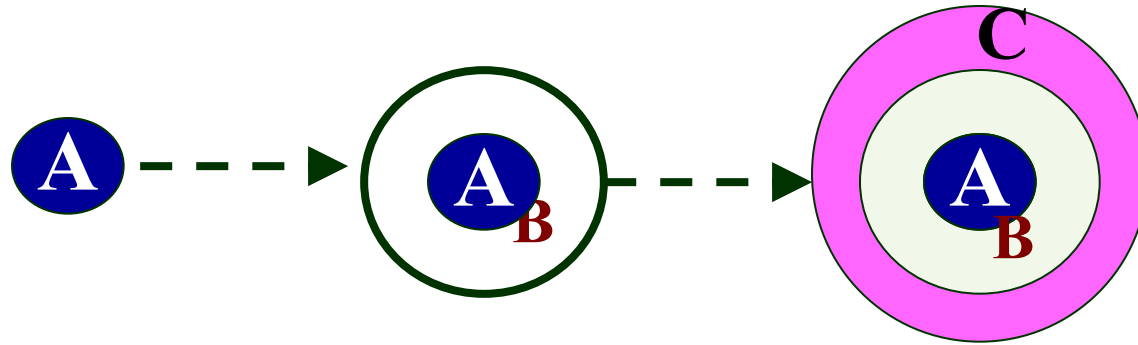
Evolutionary Model

- **Evolutionary 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.**

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.**
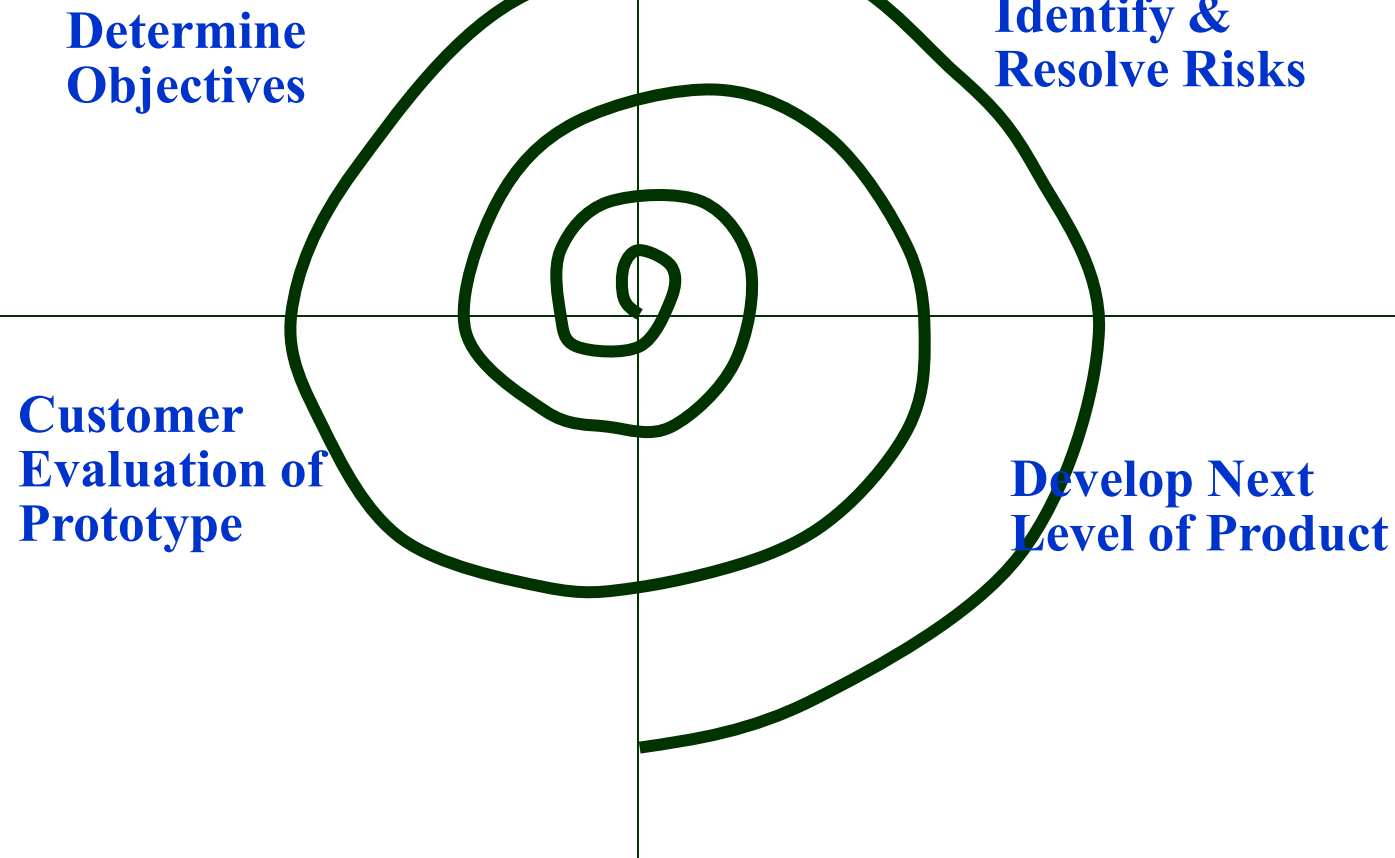
Spiral Model

- **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.)

Determine Objectives

Identify & Resolve Risks

Customer Evaluation of Prototype

Develop Next Level of Product

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

Spiral Model (CONT.)

- **<u>Development and Validation</u> (Third quadrant):**
  - develop and validate the next level of the product.

- **<u>Review and Planning</u> (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.

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.