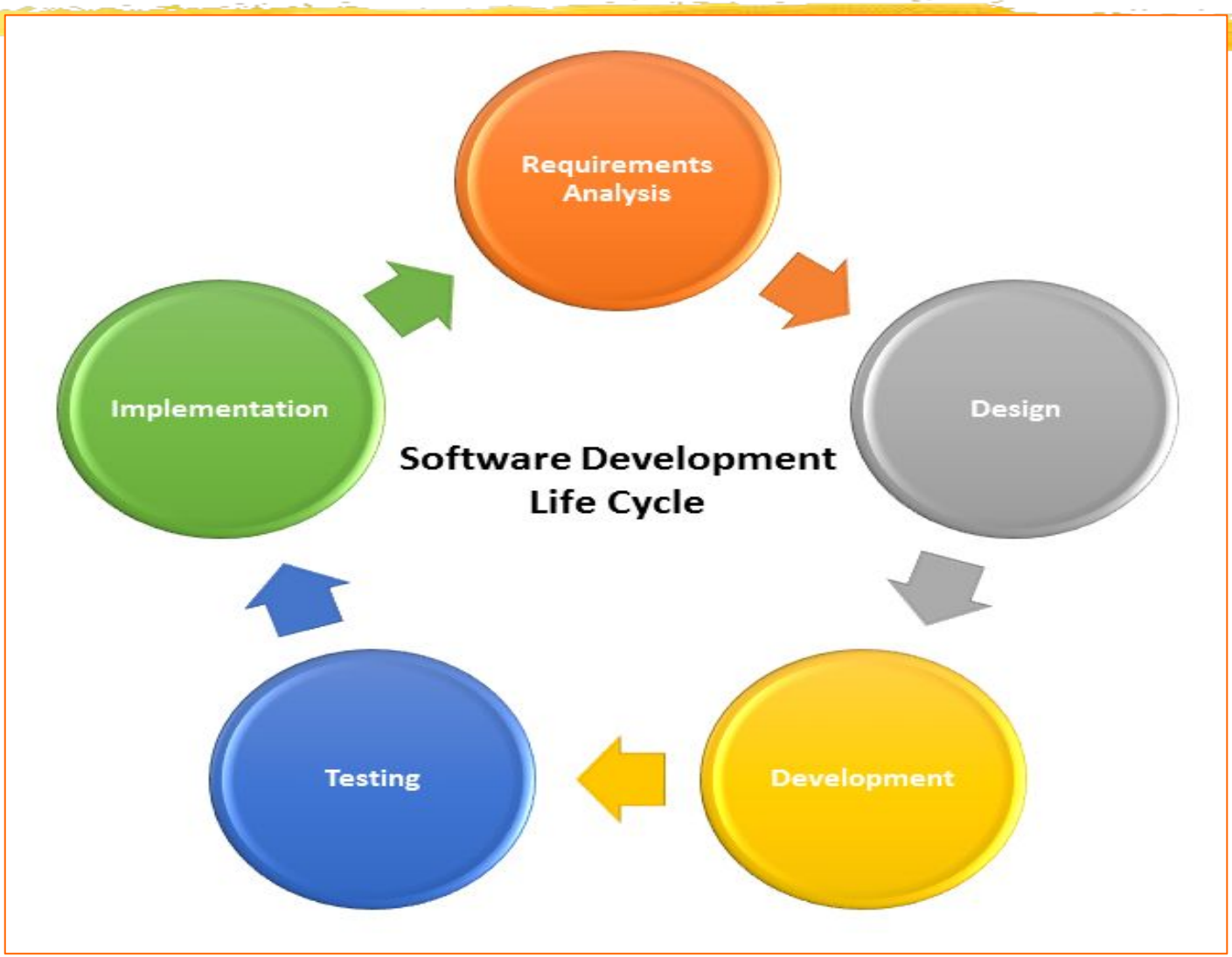
Software Life Cycle

Software Life Cycle

- A software life cycle model (also called process model) is a **descriptive and diagrammatic representation** of the software life cycle.
- Represents all the **activities** required to make a software product transit through its life cycle phases.
- Captures the **order in which these activities** are to be undertaken.
- In other words maps the different activities performed on a software product from its inception to retirement.

Activities - example



- Different life cycle models may map the basic development activities to phases in different ways.

- Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models.

- During any life cycle phase, more than one activity may also be carried out.

- For example, the **design phase** might consist of the **structured analysis** activity followed by the **structured design activity.**

Need for a software life cycle model

- Development team must identify a suitable life cycle model for the particular project and then adhere to it.
- Without using of a particular life cycle model the development of a software product would not be in a **systematic and disciplined manner**.
- When a software product is being developed by a team there must be a clear understanding among team members about **when and what to do.**
- Otherwise it would lead to chaos and project failure.

- Suppose a software development problem is divided into several parts and assigned to the team members.
- From then on, suppose the team members are allowed the freedom to develop the parts assigned to them in whatever way they like.
- It is possible that one member might start writing the code for his part,
- Another might decide to prepare the test documents first,
- And some other engineer might begin with the design phase of the parts assigned to him.
- **This would be one of the perfect recipes for project failure.**

- A software life cycle model defines **entry and exit criteria for every phase.**

- A phase can start only if its **phase-entry criteria** have been satisfied.

- So without software life cycle model the **entry and exit criteria for a phase cannot be recognized.**

- Without software life cycle models it becomes difficult for software project managers to monitor the progress of the project.

Different software life cycle models

- Many life cycle models have been proposed so far.
- Each of them has some advantages as well as some disadvantages.
  - Classical Waterfall Model
  - Iterative Waterfall Model
  - Prototyping Model
  - Evolutionary Model
  - Spiral Model

- Identify the problem one would face, if he tries to develop a large software product without using software engineering principles.
- Identify the two important techniques that software engineering uses to tackle the problem of exponential growth of problem complexity with its size.
- State five symptoms of the present software crisis.
- Identify at least four basic characteristics that differentiate a simple program from a software product.
- Identify at least two advantages of using high-level languages over assembly languages.
- State at least five advantages of object-oriented design techniques.
- State at least three differences between the exploratory style and modern styles of software development.

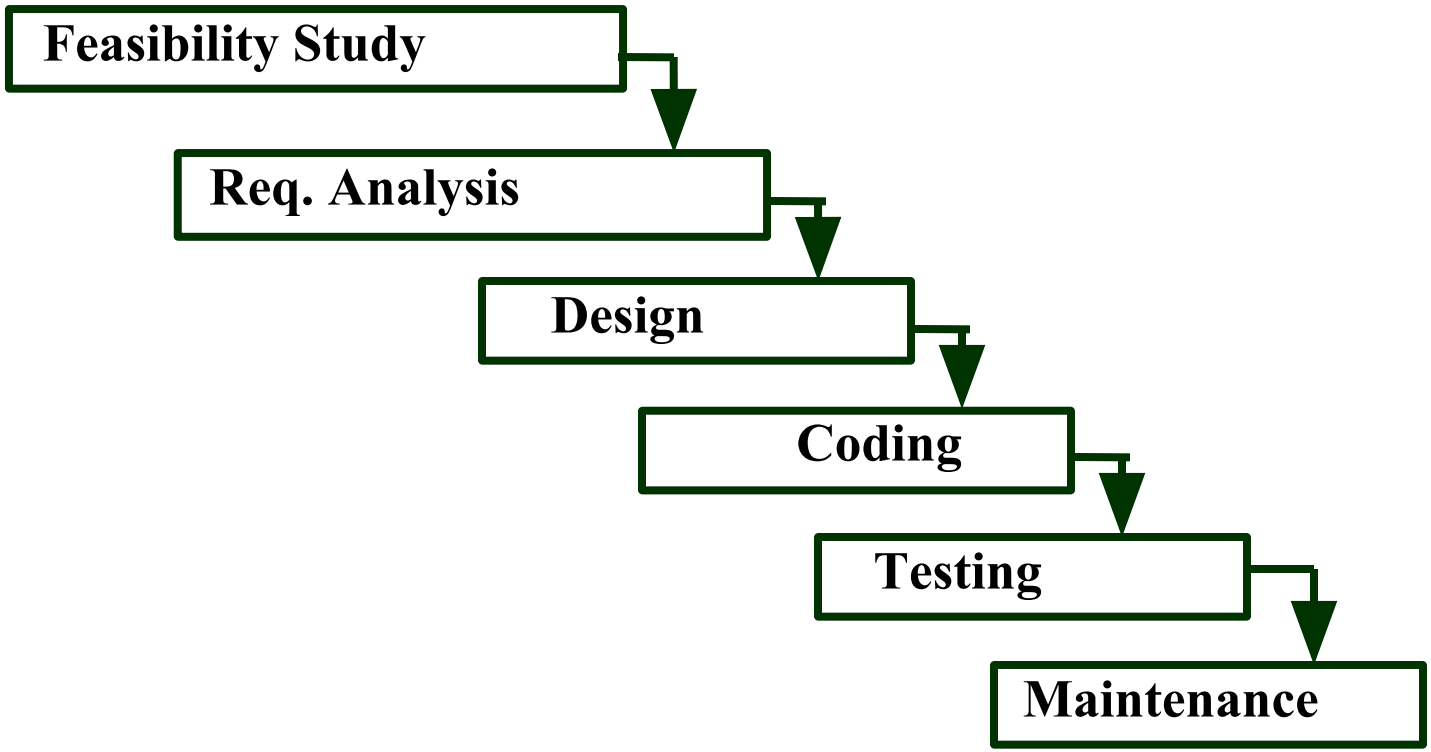
Model-1

Classical Waterfall Model

Classical Waterfall Model

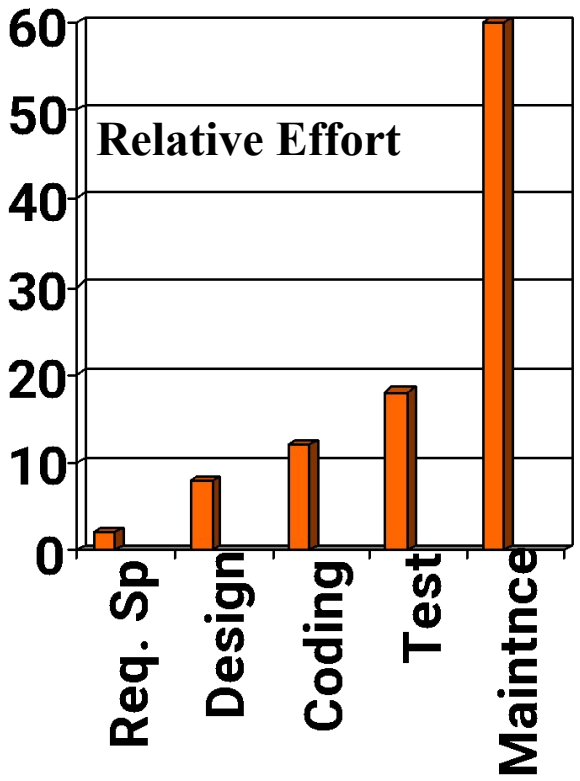
- Classical waterfall model divides life cycle into phases:
  - feasibility study,
  - requirements analysis and specification,
  - design,
  - coding and unit testing,
  - integration and system testing,
  - maintenance.

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.



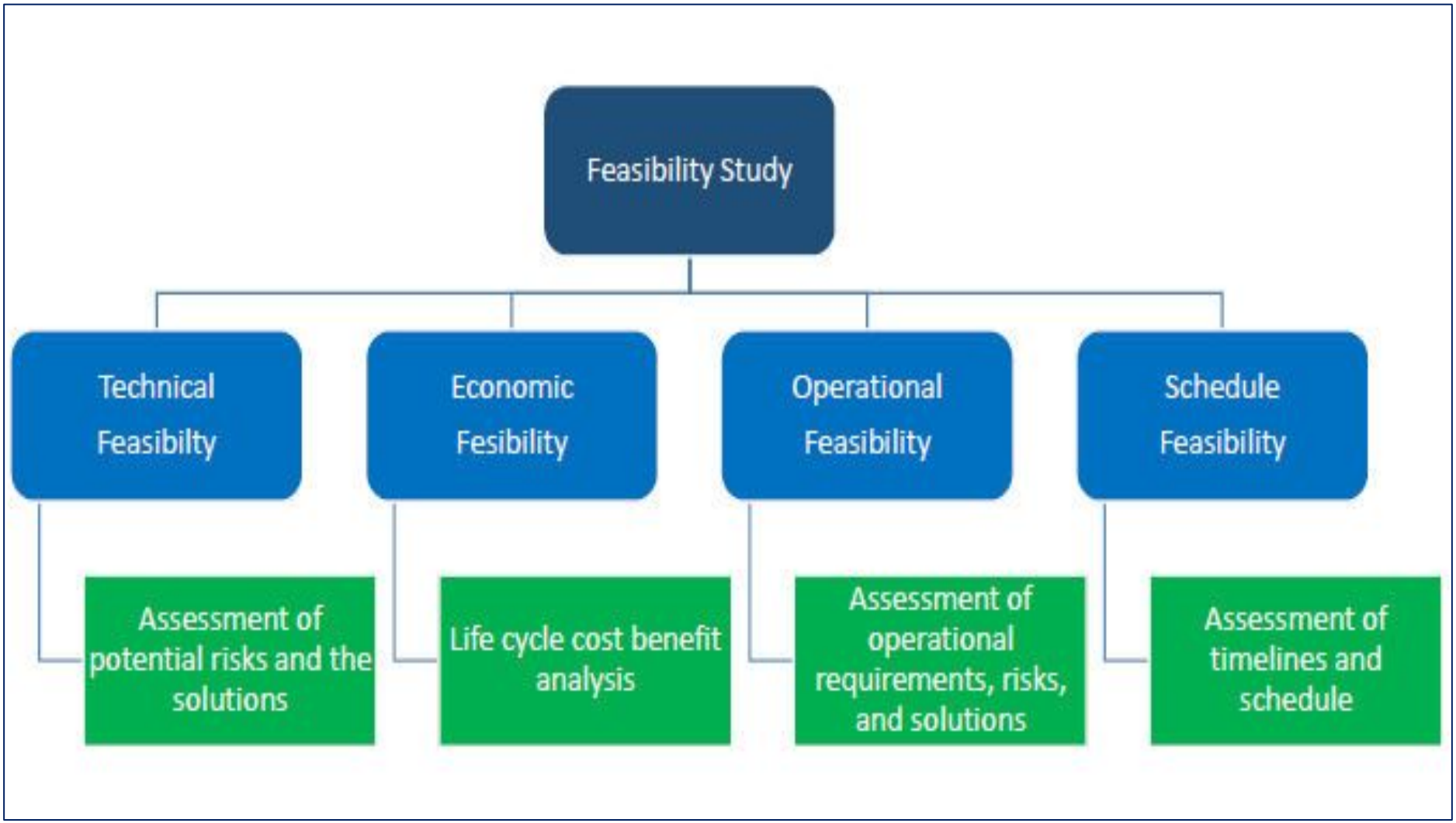
Classical Waterfall Model
(CONT.)

- **Most organizations usually define:**
  - standards on the outputs (deliverables) produced at the end of every phase
  - entry and exit criteria for every phase.
- **They also prescribe specific methodologies for:**
  - specification,
  - design,
  - testing,
  - project management, etc.

a) Feasibility Study

- **Main aim of feasibility study: determine whether developing the product**
  - **financially worthwhile**
  - **technically feasible.**
- **First roughly understand what the customer wants by:**
  - **Data** : study different **input data** to the system and **output data** to be produced by the system
  - **Process :**what kind of processing is needed to be done on these data and they look at the various constraints on the behaviour of the system.
  - After an overall understanding of the problem they investigate the **different solutions** that are possible.
  - **Examine each of the solutions** in terms of resources required, cost of development and development time for each solution.

Feasibility Study

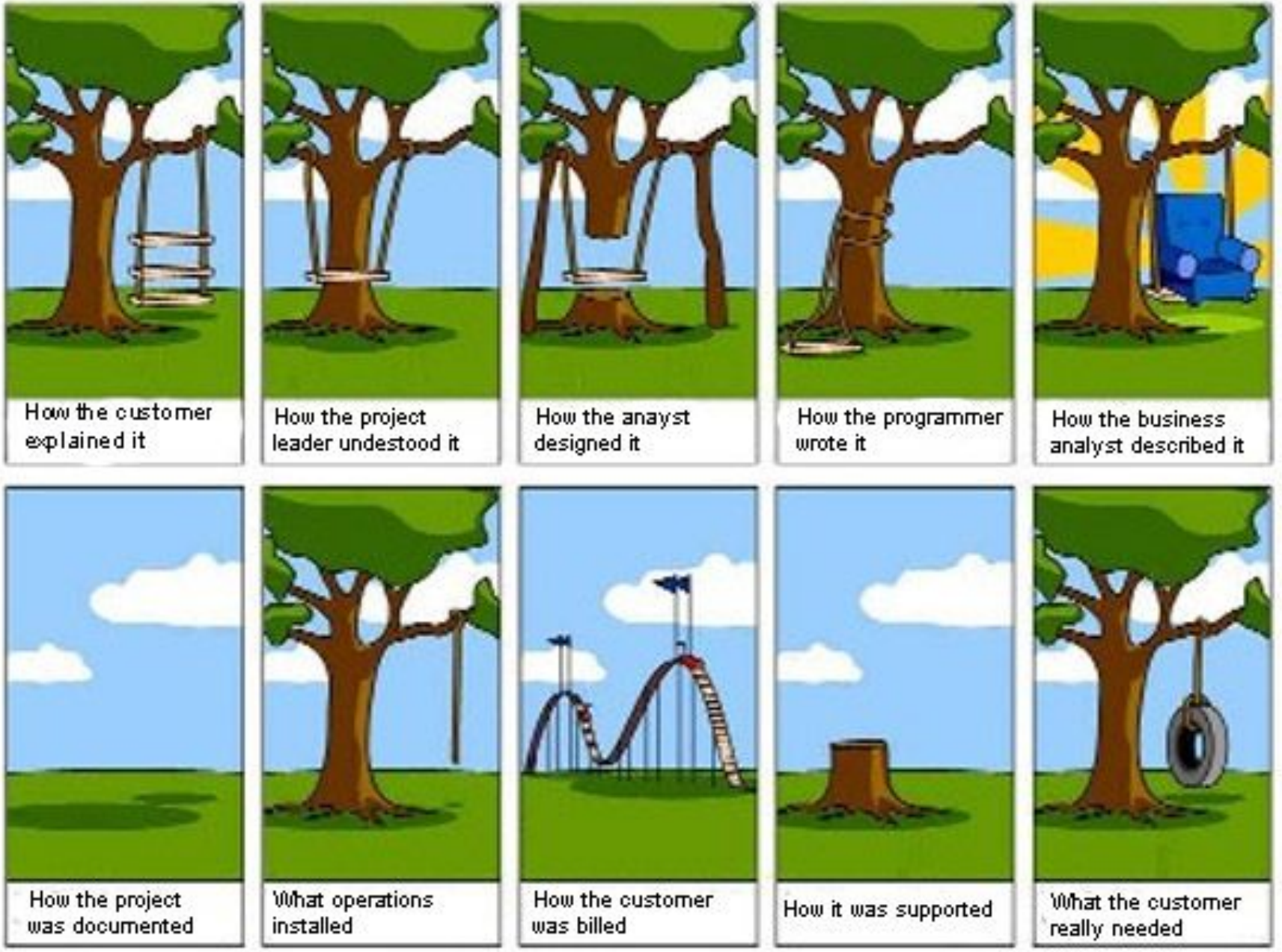


Activities during Feasibility Study

- **Perform a cost/benefit analysis:**
  - Based on this analysis pick the best solution and determine whether the solution is feasible financially and technically.
  - Check whether the customer budget would meet the cost of the product and whether they have sufficient technical expertise in the area of development.

b) 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.**


How the customer explained it
How the project leader undestood it
How the anayst designed it
How the programmer wrote it
How the business analyst described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

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 Process

- **One-on-One Interviews :** These can be users who interact with the current or new system, management, project financers or anyone else that would be involved in the system. **C**ontain open and close ended questions.
- **Group Interviews :** Important to note which issues are generally agreed upon, and which issues differ.
- **Questionnaires/Surveys :** This is especially helpful when stakeholders are spread out geographically
- **User Observation:** For optimal results, the consultant should schedule three different periods of observation: low, normal, and peak times. This may prove helpful in because the user may interact with the system differently during different times.
- **Analyzing Existing Documents:** Reviewing the current process and documentation can help the analyst understand the business, or system, and its current situation.
- **Joint Application Design/JAD**

- **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.

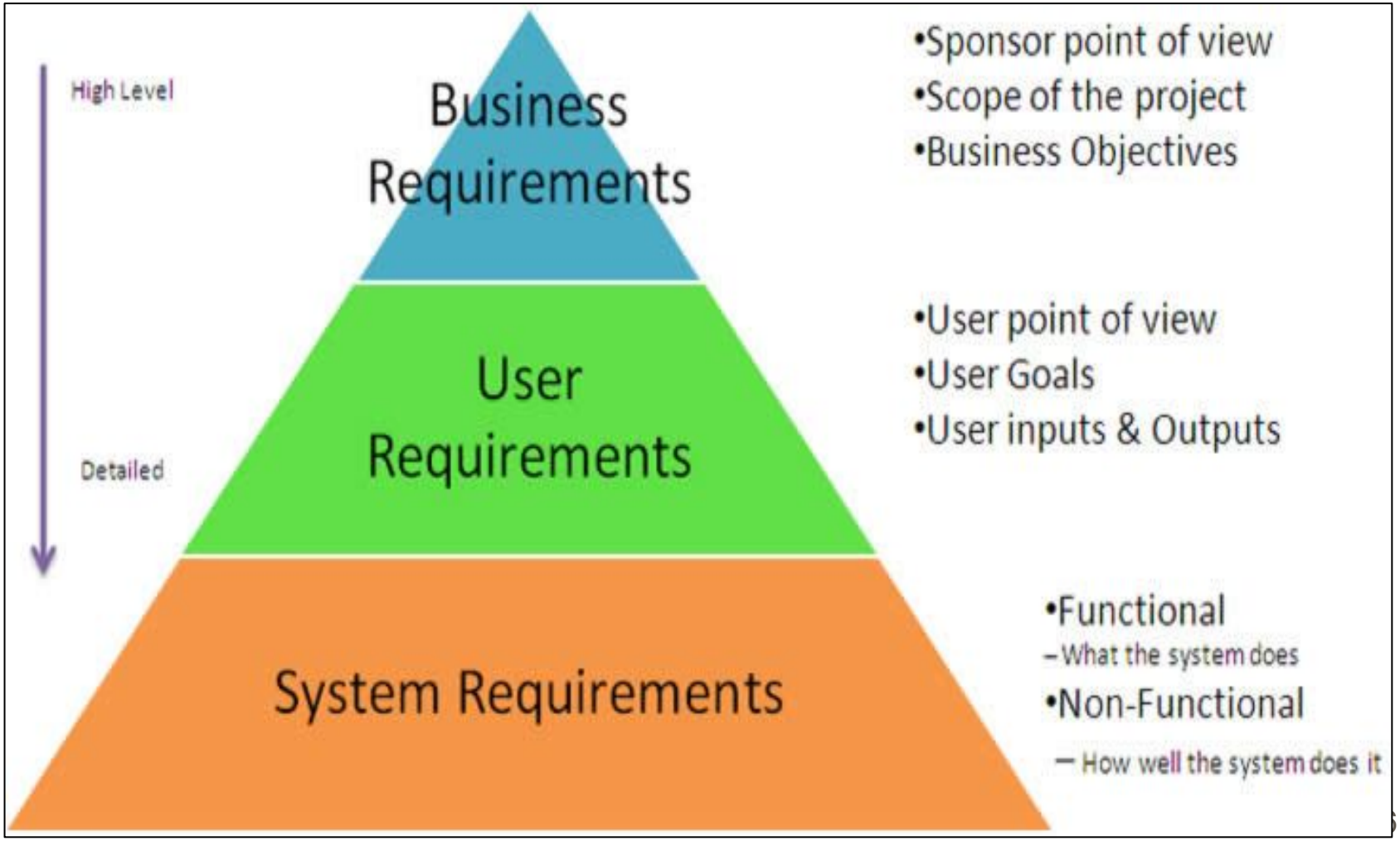
Requirements Analysis (CONT.)

- **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.**

Requirements types



c) 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:
  - a) traditional approach,
  - b) object oriented approach.

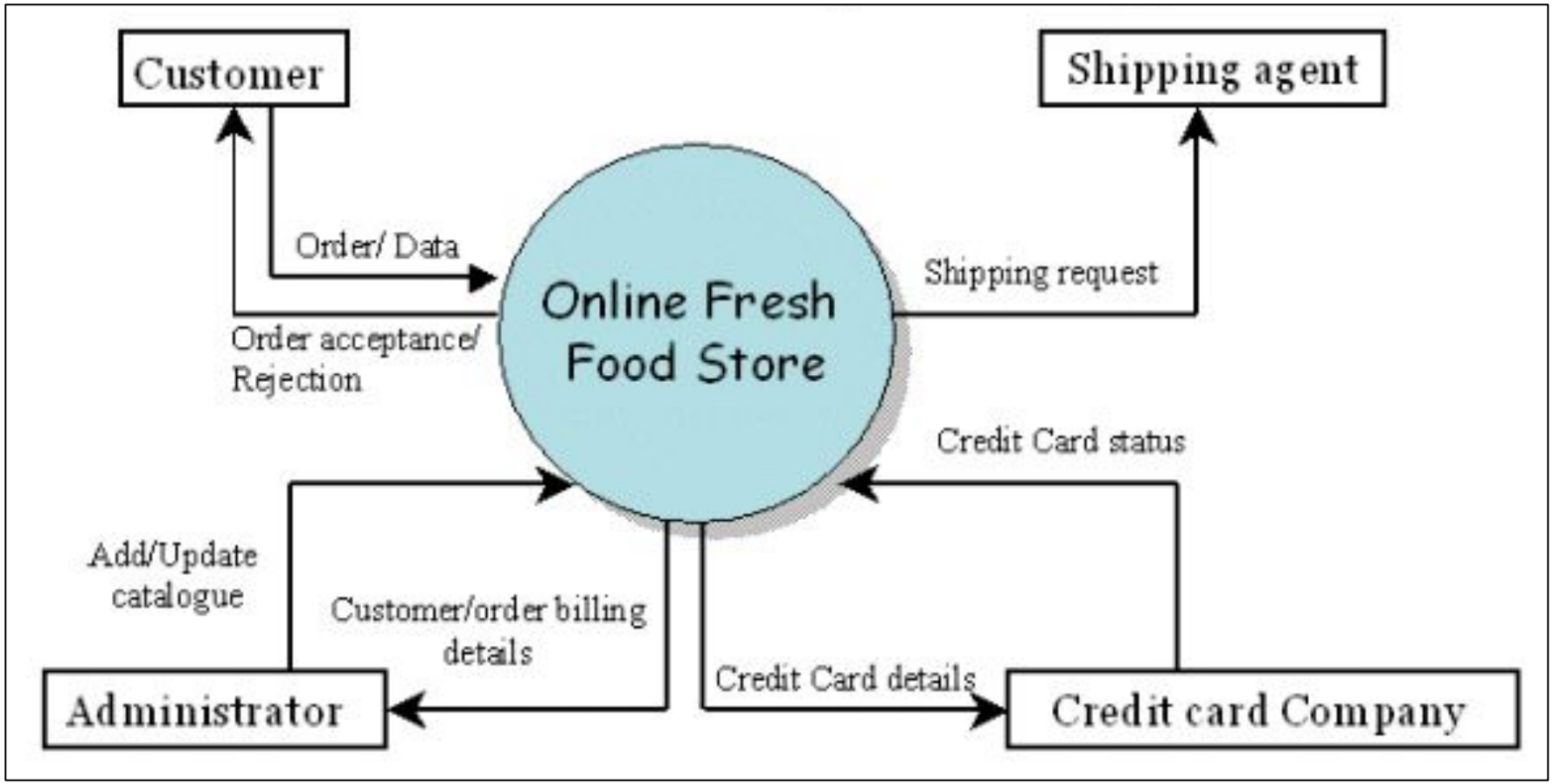
a) Traditional approach – Design Phase

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

1. 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.
- **Carried out using Data flow diagrams (DFDs).**

Data flow diagram example



Traditional approach – Design Phase

2. Structured Design

After structured analysis, carry out **structured design**:

- **architectural design** (or high-level design)
- **detailed design** (or low-level design).

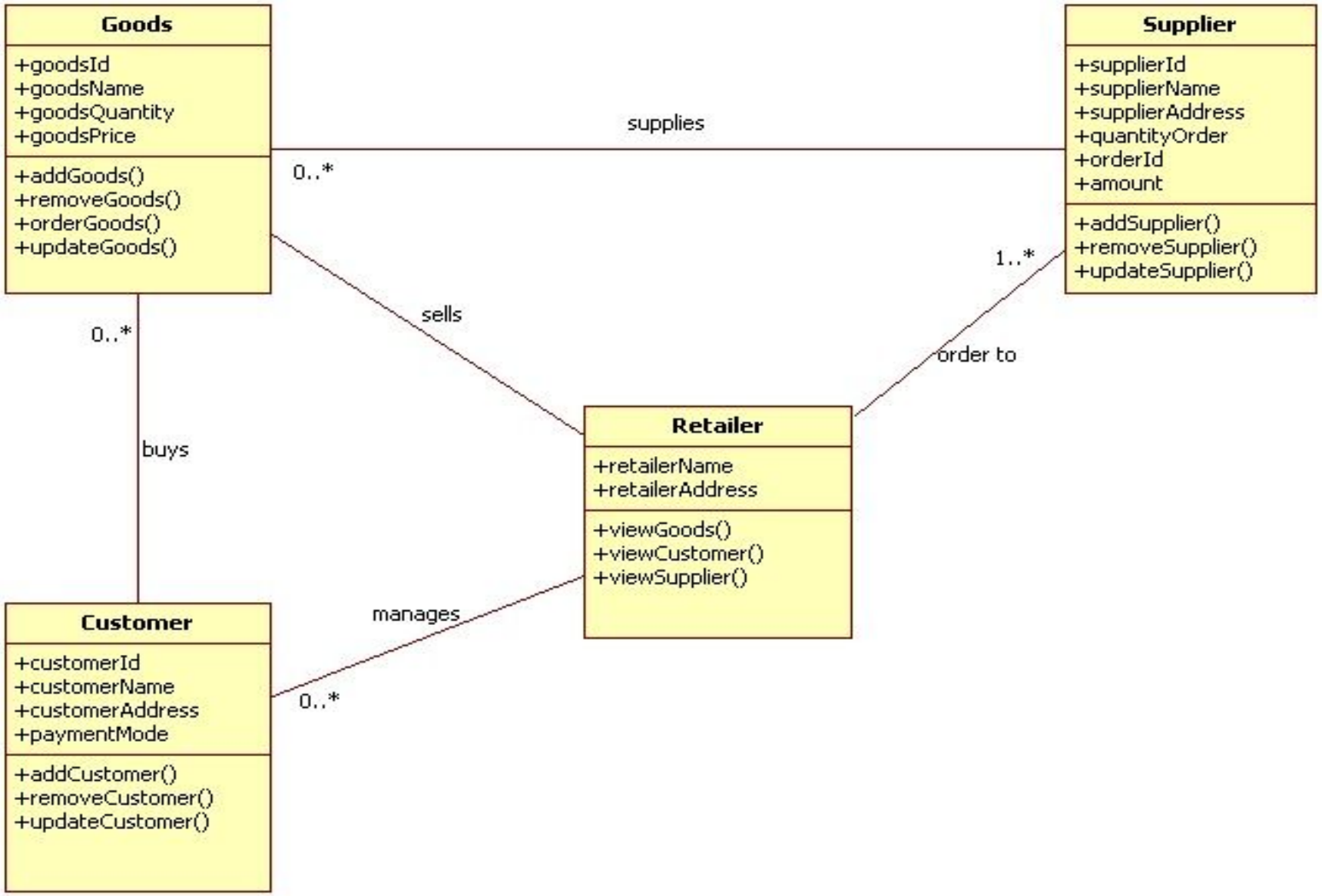
- **High-level design:**
  - **decompose the system into *modules*,**
  - **represent invocation relationships among the modules.**
- **Detailed design:**
  - **different modules designed in greater detail:**
    - **data structures and algorithms for each module are designed.**

b) 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 structure
  - further refined to obtain the detailed design.

- OOD has several advantages:
  - lower development effort,
  - lower development time,
  - better maintainability.

example of object oriented design



d) Implementation

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

- 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.
- **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.

e) 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.**

f) 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.**
- 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.

Model-2

Iterative Waterfall Model

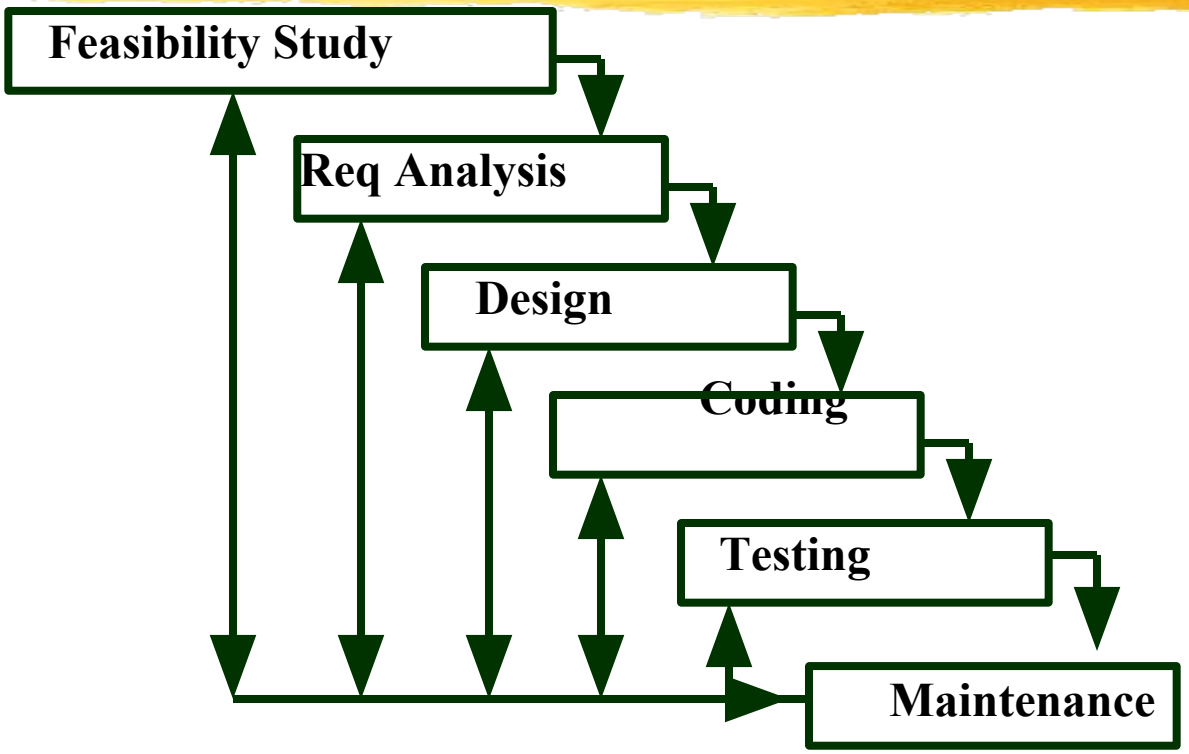
Iterative Waterfall Model

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

- 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

- **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

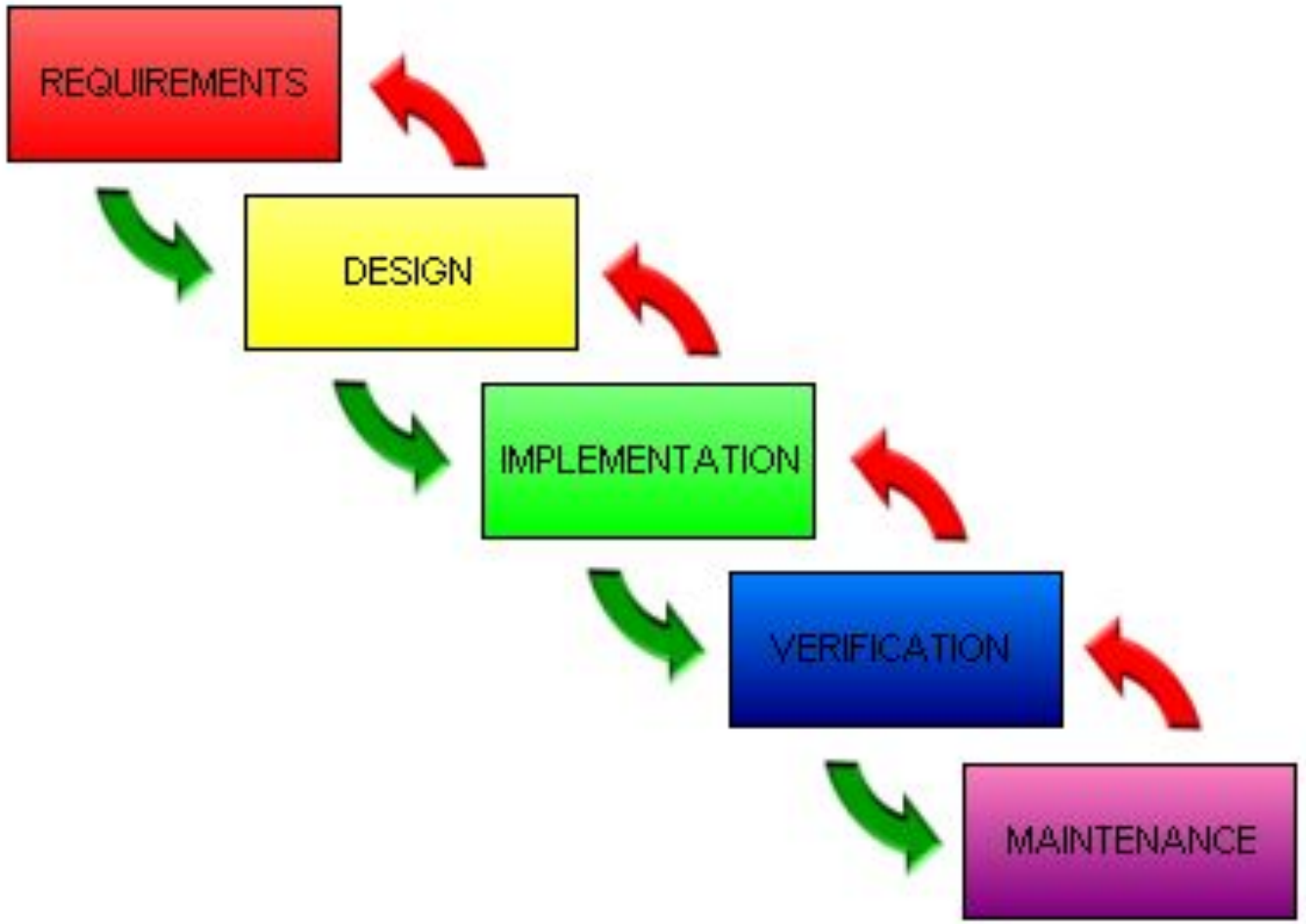
- **Reason:** rework must be carried out not only to the design but also to code and test phases.

- 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.

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

Iterative Waterfall Model



Model-3

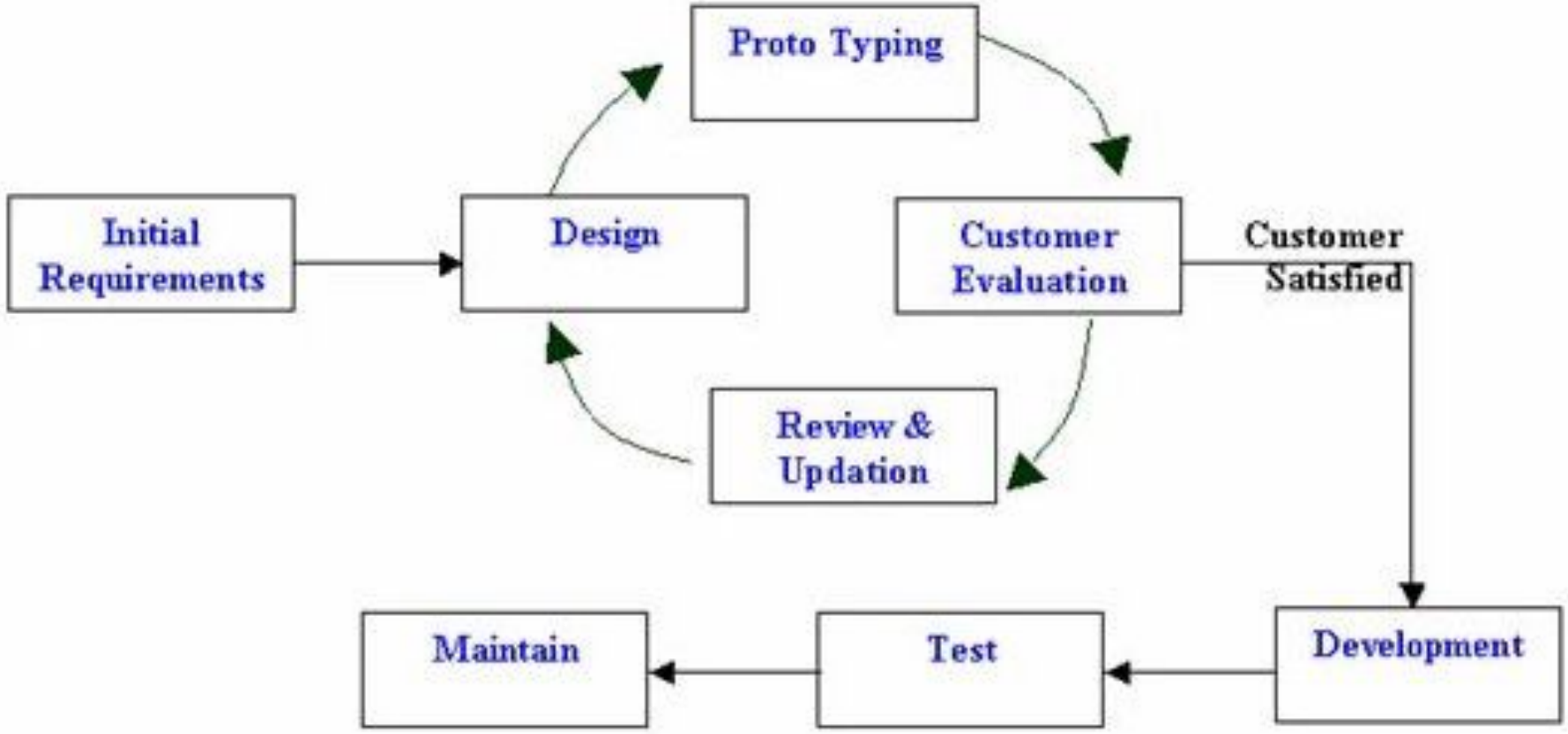
Prototyping Model

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.**

prototype model



Proto Type Model

Reasons for developing a prototype

- 1) Illustrate to the customer:
  - input data formats, messages, reports, or interactive dialogs.

- 2) 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.)

- **3)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.**

- **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.)

- **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.

Model-4

Evolutionary Model

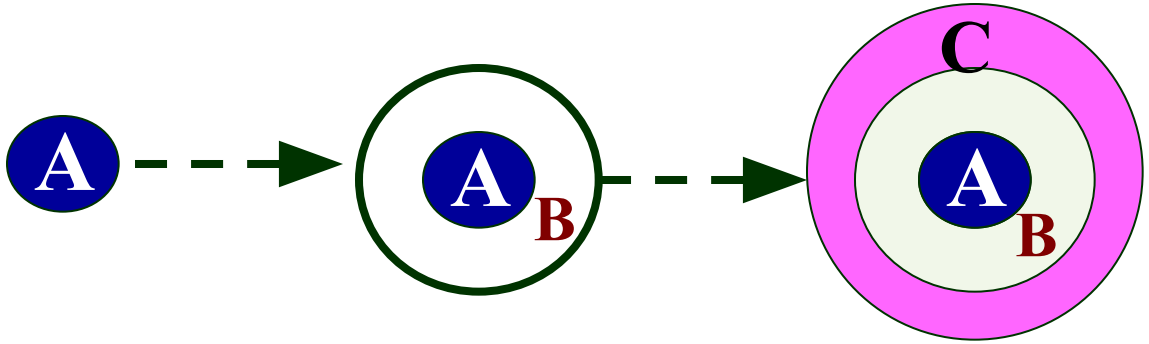
Advantages of Evolutionary Model

- **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.**

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.**
- **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.)



Disadvantages of Evolutionary Model

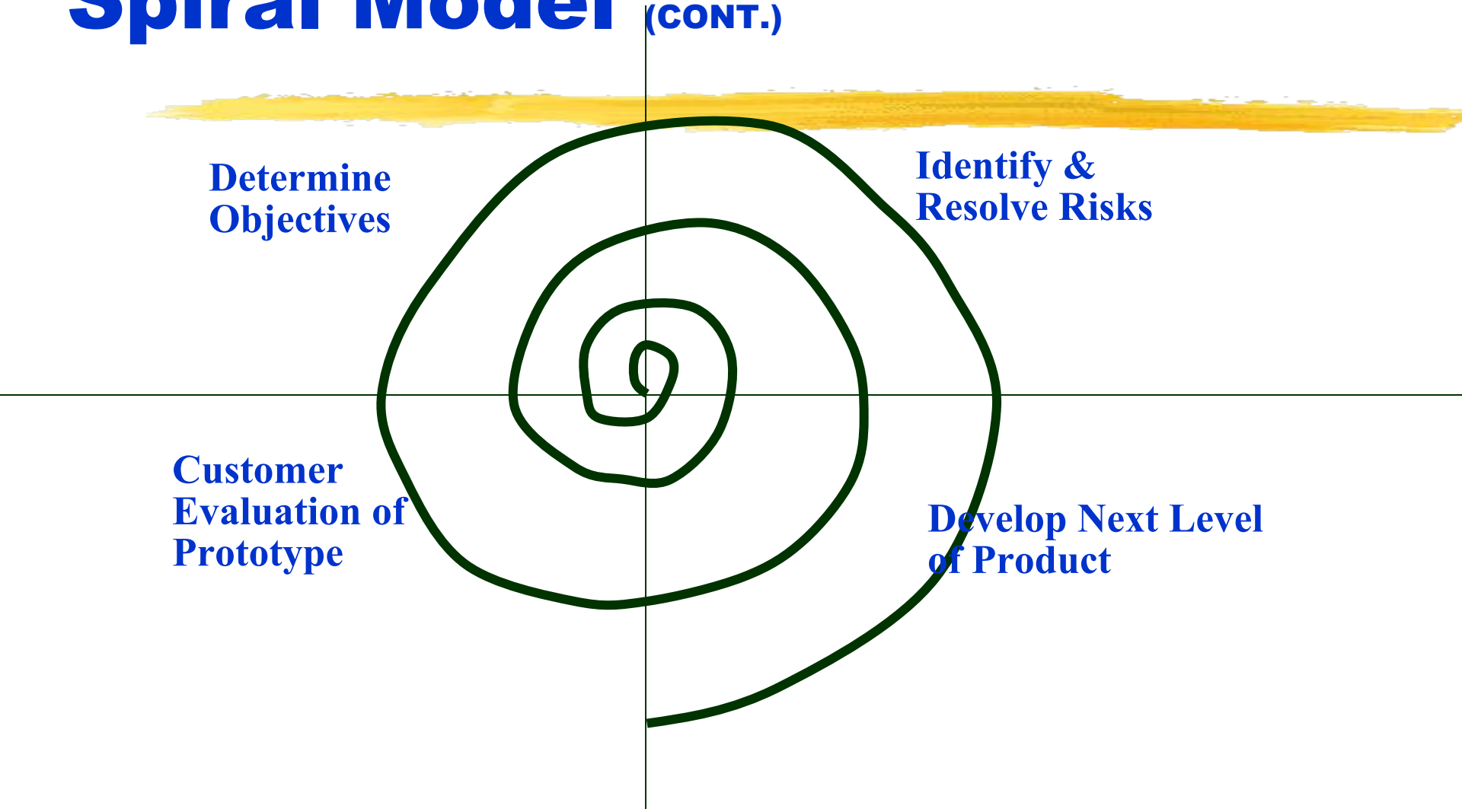
- **Often, difficult to subdivide problems into functional units:**
  - **which can be incrementally implemented and delivered.**
  - **evolutionary model is useful for very large problems,**
    - **where it is easier to find modules for incremental implementation.**

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.**

- **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 (CONT.)



Spiral Model

- **Each loop of the spiral represents a phase of the software process:**
  - **the innermost loop might be concerned with system feasibility,**
  - **the next loop with system requirements definition,**
  - **the next one with system design, and so on.**

- **The exact number of loops in the spiral is not fixed.**

Spiral Model (CONT.)

- The team must decide:
  - how to structure the project into phases.

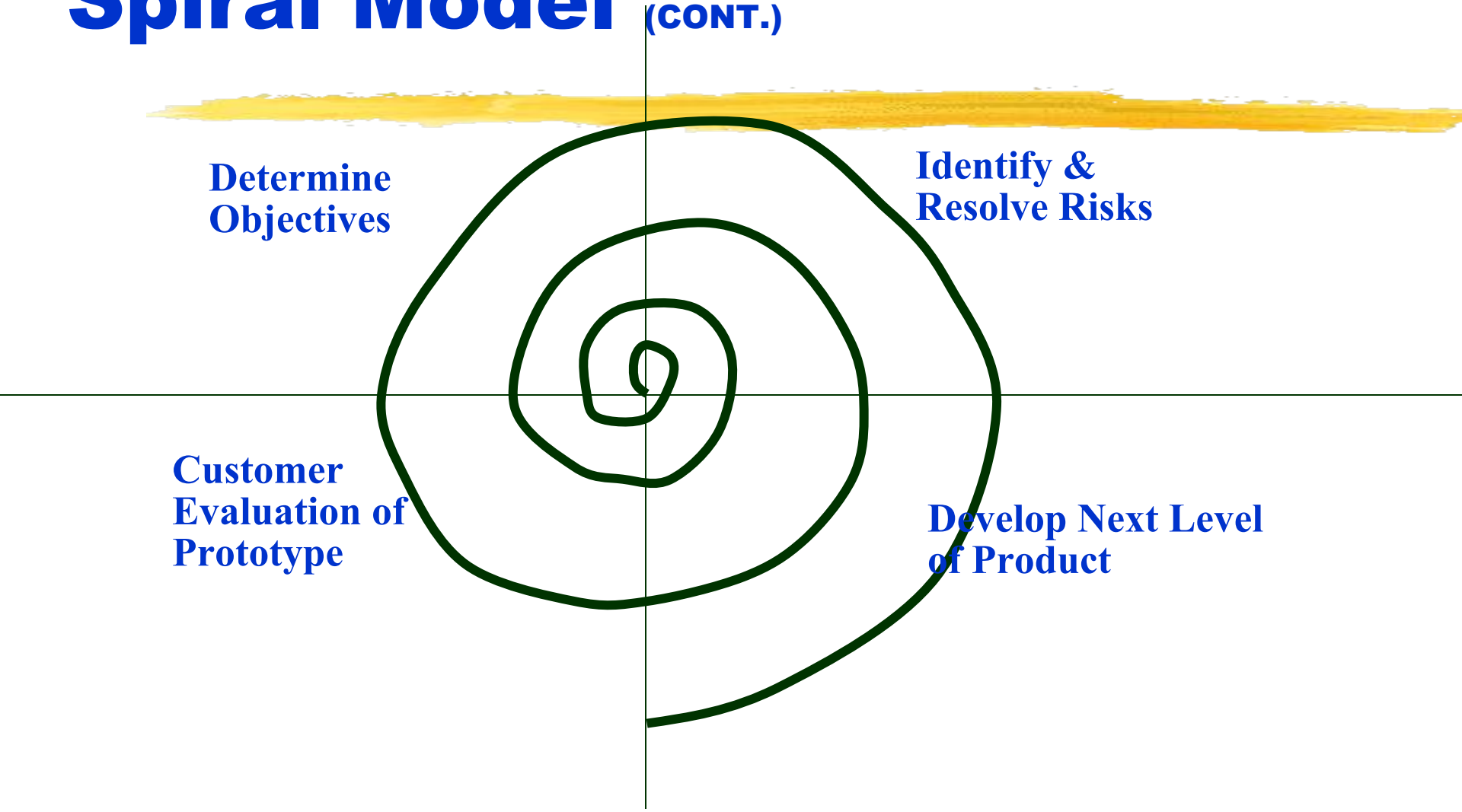
- Start work using some generic model:
  - add extra phases
    - for specific projects or when problems are identified during a project.

- Each loop in the spiral is split into four sectors (quadrants).

Model-5

Spiral Model

Spiral Model (CONT.)



Objective Setting (First Quadrant)

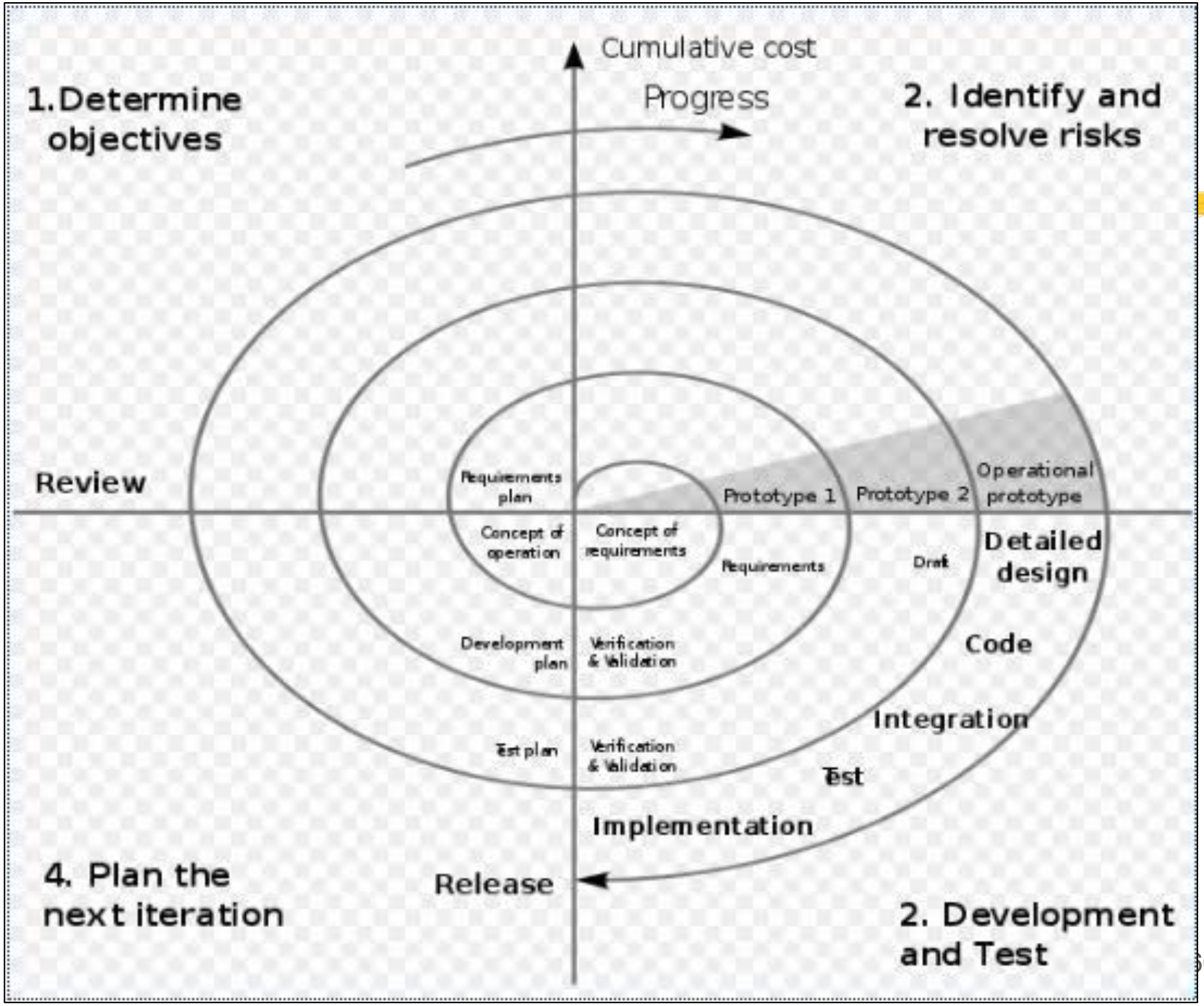
- Identify objectives of the phase,
- Examine the risks associated with these objectives.
  - Risk:
    - any adverse circumstance that might hamper successful completion of a software project.
- Find alternate solutions possible.

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.


Cumulative cost
Progress
1. Determine objectives
2. Identify and resolve risks
Review
Requirements plan
Prototype 1
Prototype 2
Operational prototype
Concept of operation
Concept of requirements
Requirements
Draft
Detailed design
Development plan
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
4. Plan the next iteration
Release
2. Development and Test

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.

Advantages of Spiral Model

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the **software life cycle**.

Disadvantages

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

Circumstances to use spiral model

- The spiral model is called a meta model since it encompasses all other life cycle models.
- Risk handling is inherently built into this model.
- The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks.
- However, this model is much more complex than the other models **– this is probably a factor deterring its use in ordinary projects.**

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.

Model-5

Rapid Application Development Model

RAD

- RAD model is Rapid Application Development model.
- It is a type of **incremental model**.
- In RAD model the components or functions are developed in parallel as if they were mini projects.
- The developments are time boxed, delivered and then assembled into a working prototype.
- This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

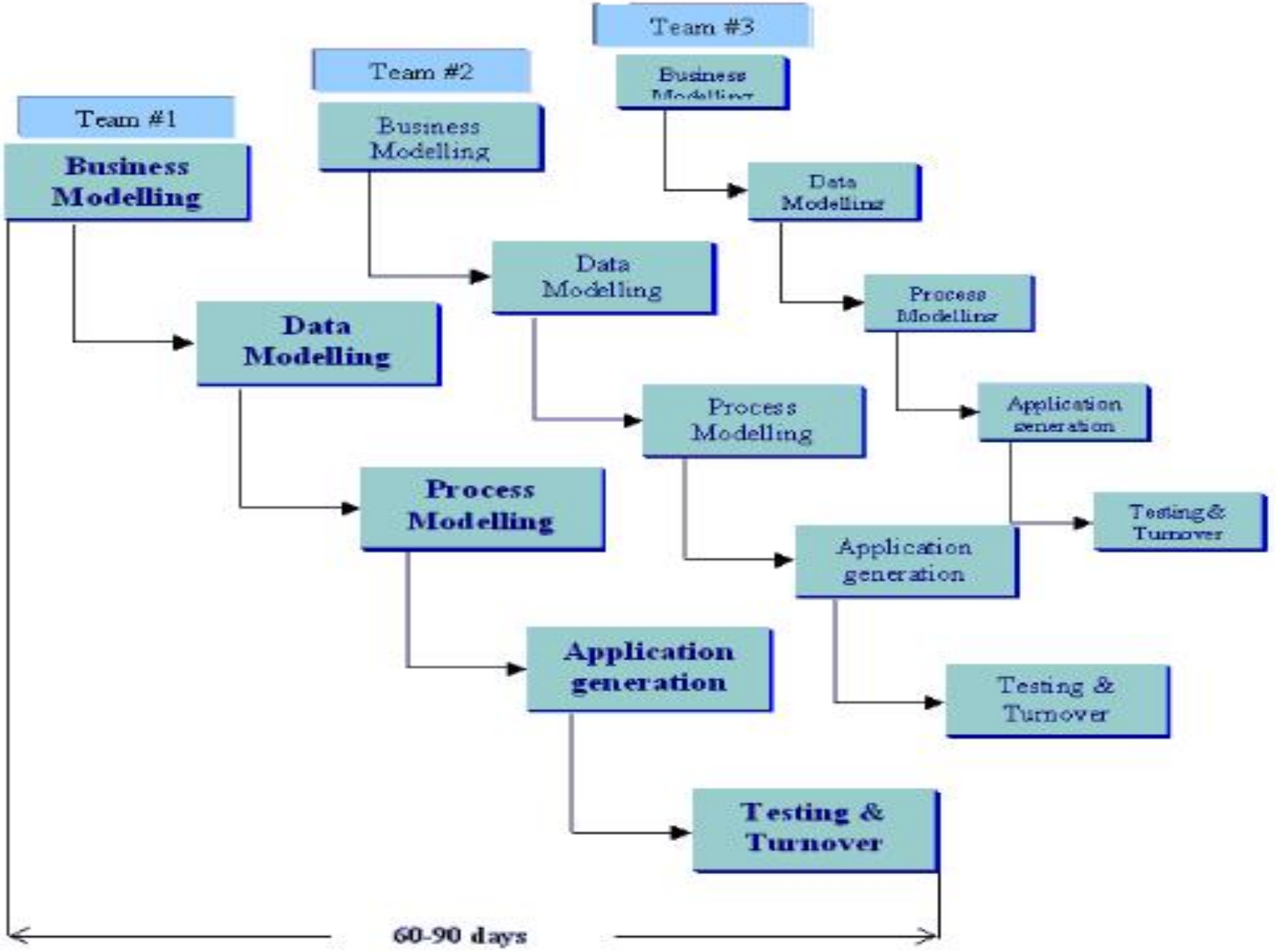


Figure 1.5 – RAD Model

- The phases in the rapid application development (RAD) model are:
  - **Business Modeling:** In this phase of development business model should be designed based on the information available from different business activities.
  - Before start the development there should be a complete picture of business process functionality.
  - **Data Modeling:** Once the business modeling phase over and all the business analysis completed, all the required and necessary data based on business analysis are identified in data modeling phase.
  - **Process Modeling:** All the data identified in data modeling phase are planned to process or implement the identified data to achieve the business functionality flow. In this phase all the data modification process is defined.

- **Application Modeling:** In this phase application id developed and coding completed. With help of automation tools all data implemented and processed to work as real time.

- **Testing and turnover:** All the testing activates are performed to test the developed application.

- **Advantages of RAD Model:**
  - Fast application development and delivery.
  - Least testing activity required.
  - Visualization of progress.
  - Less resources required.
  - Review by the client from the very beginning of development so very less chance to miss the requirements.
  - Very flexible if any changes required.
  - Cost effective.
  - Good for small projects.

- **Disadvantages of RAD Model:**
  - High skilled resources required.
  - On each development phase client's feedback required.
  - Automated code generation is very costly.
  - Difficult to manage.
  - Not a good process for long term and big projects.
  - Proper modularization of project required.

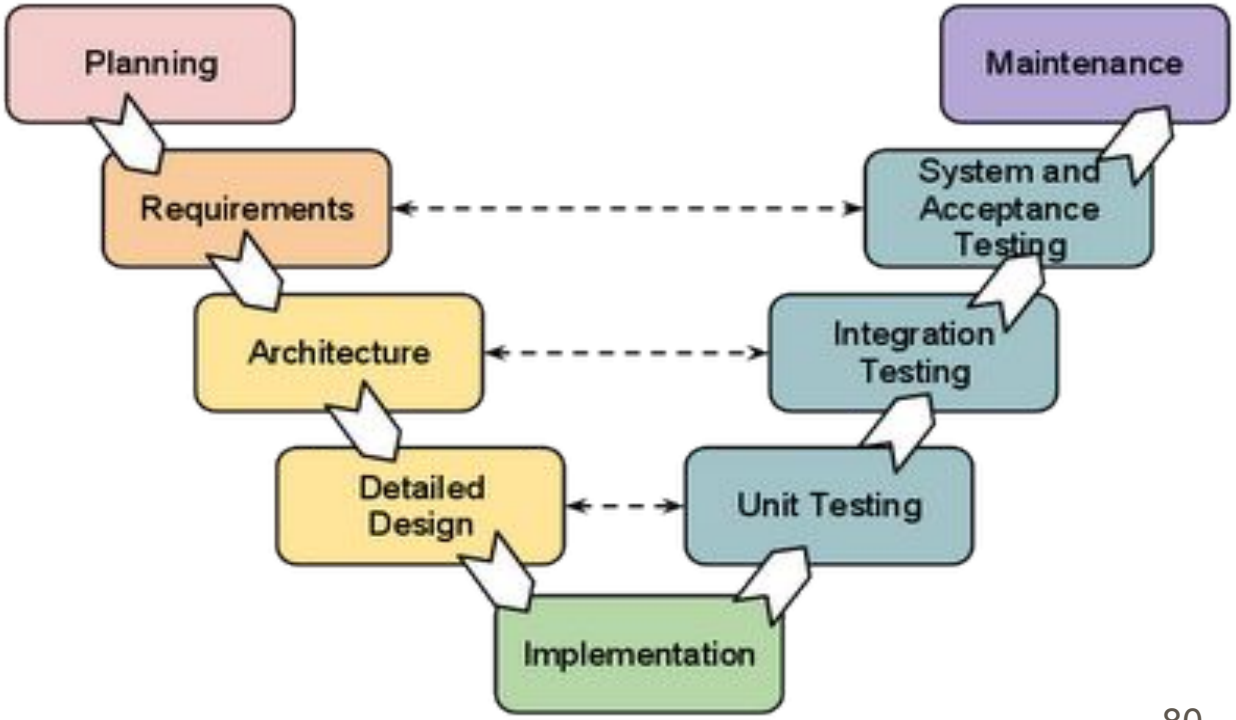
Model-6

V shaped Model

V-Shaped Model

- It is an extension for waterfall model, Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape.

- The major difference between v-shaped model and waterfall model is the early test planning in v-shaped model.



- The usage
  - Software requirements clearly defined and known
  - Software development technologies and tools is well-known

- **Advantages**
  - Simple and easy to use.
  - Each phase has specific deliverables.
  - Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
  - Works well for where requirements are easily understood.
  - Verification and validation of the product in early stages of product development

Disadvantages

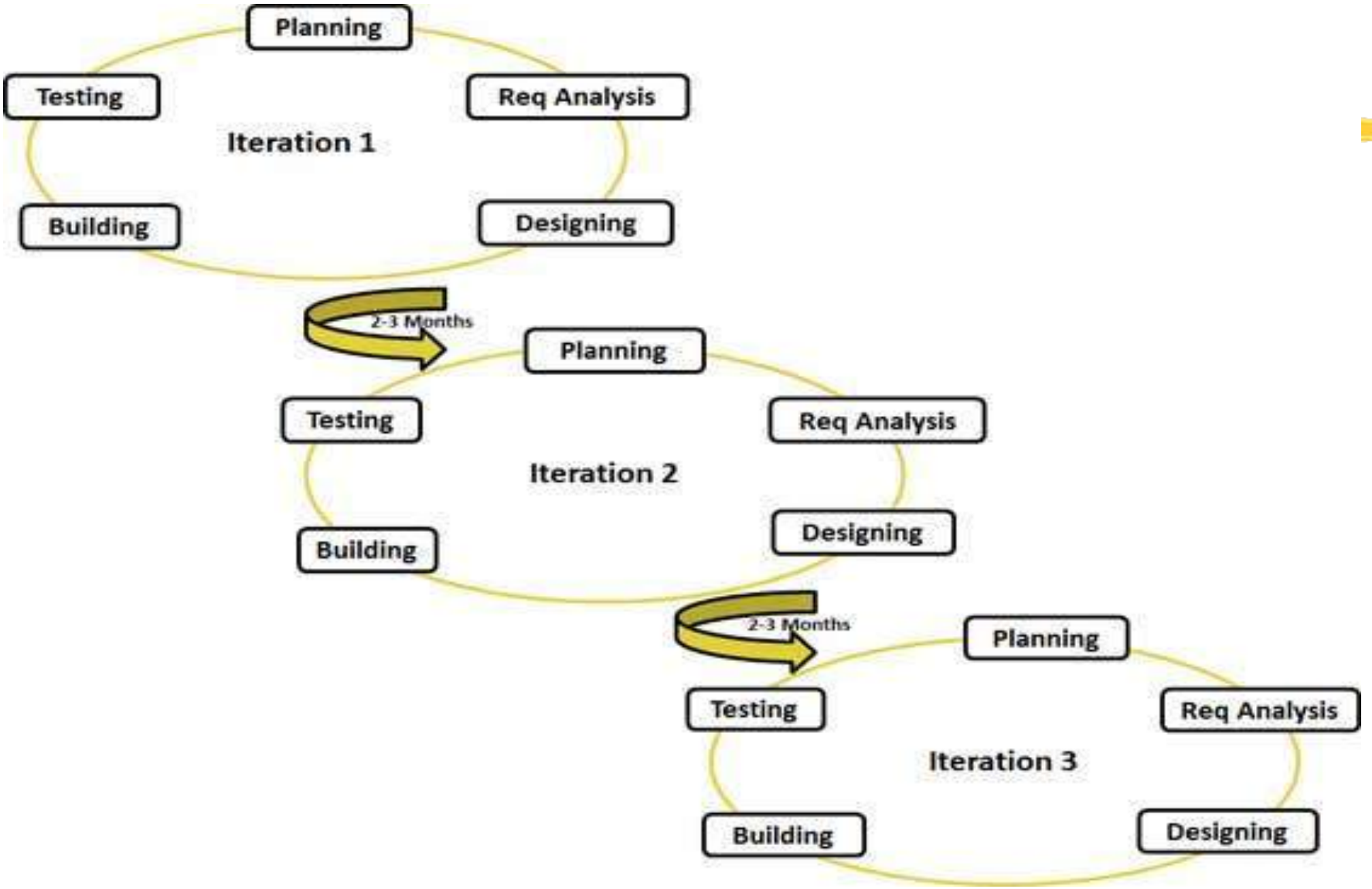
- Very inflexible, like the waterfall model.
- Little flexibility and adjusting scope is difficult and expensive.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- Model doesn't provide a clear path for problems found during testing phases.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

Model-7

Agile Model

Agile Model

- **Combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.**
- Agile Methods break the product into small incremental builds.
- These builds are provided in iterations. **Each iteration typically lasts from about one to three weeks.**
- Every iteration involves **cross functional teams** working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
- At the end of the **iteration a working product is displayed to the customer** and important stakeholders.


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing

Advantages

- **Is a very realistic approach** to software development
- **Promotes teamwork** and cross training.
- **Functionality can be developed rapidly and demonstrated.**
- Resource requirements are minimum.
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.

Disadvantages

- Not suitable for **handling complex dependencies.**
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- **Depends heavily on customer interaction**, so if customer is not clear, team can be driven in the wrong direction.
- There is very **high individual dependency**, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

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.

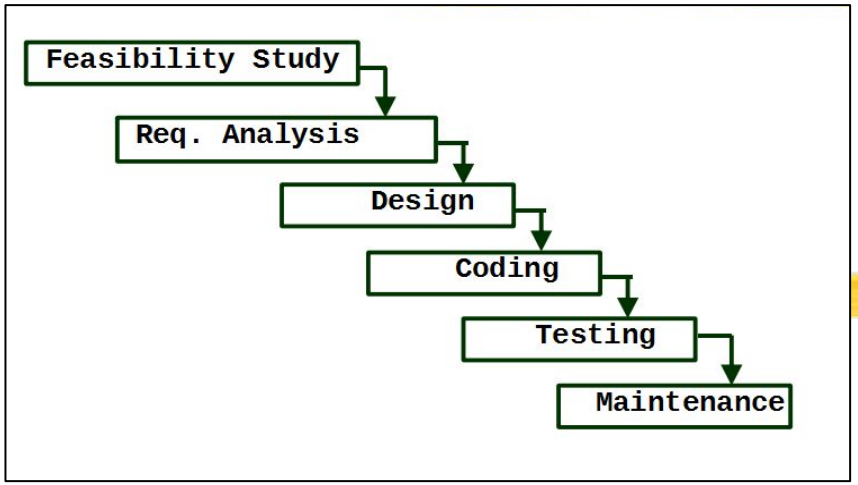
- Identify the definite stages through which a software product undergoes during its lifetime.
- Explain the problems that might be faced by an organization if it does not follow any software life cycle model.
- Identify two basic roles of a system analyst.
- Differentiate between structured analysis and structured design.
- Identify at least three activities undertaken in an object-oriented software design approach.

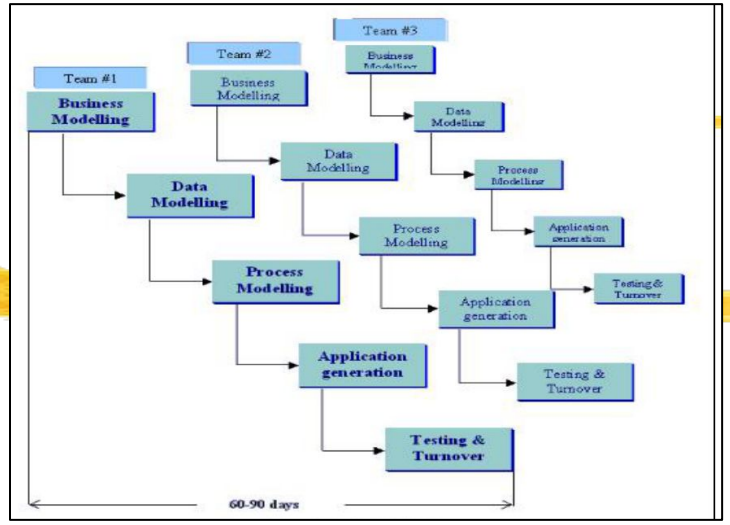
Agile vs Iterative vs Waterfall – {Process}

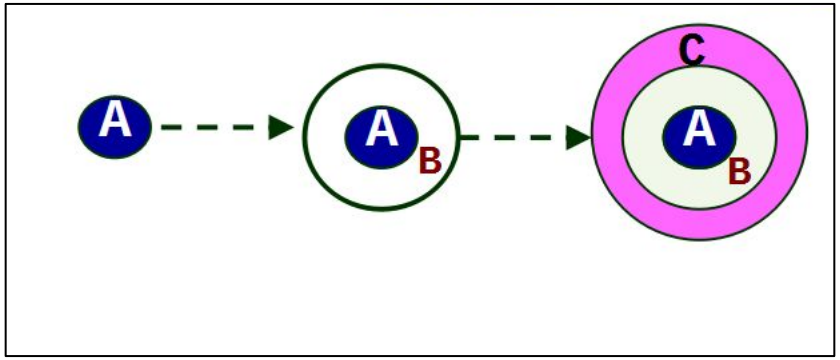
	Waterfall	Iterative (hybrid)	Agile
Quality	Quality focus changes from Analysis > Design > Code > Test	Quality focus shifts between Analysis/Design phase to Coding/Testing phase	Quality focus on all aspects of SDLC at any given time.
Quality Control	Detection & fixing during system and regression testing at the last phase of project.	Early detection & fixing in each iteration for new features. Followed by regression testing.	Early detection & fixing in each sprint followed by stabilization.
Continual Improvement (CA & PA)	Lessons learned from previous release implemented in next release	Lessons learned from previous Iteration implemented in next Iteration.	Lessons learned from previous sprint implemented in next sprint
Risk	No Risk Identification. Firefighting during testing phase.	Risk identification & mitigation in dev & test phase of each iteration.	Early identification & mitigation in every sprint.
Postmortem/ Retrospection	After every release	After every iteration/ milestone	After every sprint in retrospection meeting
Customer Feed back	At the end of the project.	At the end of every iteration	At the end of every sprint

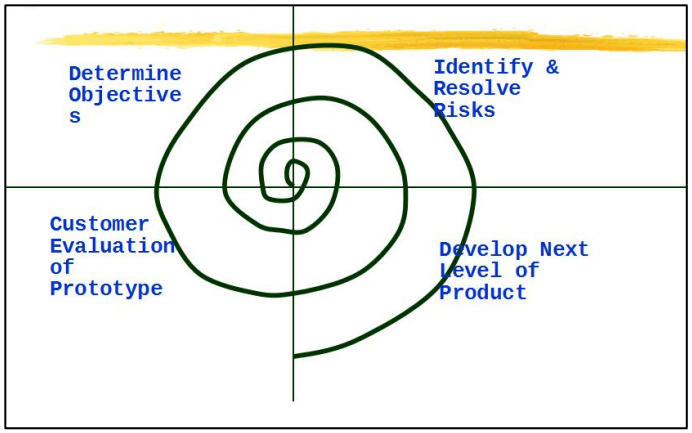
we learned

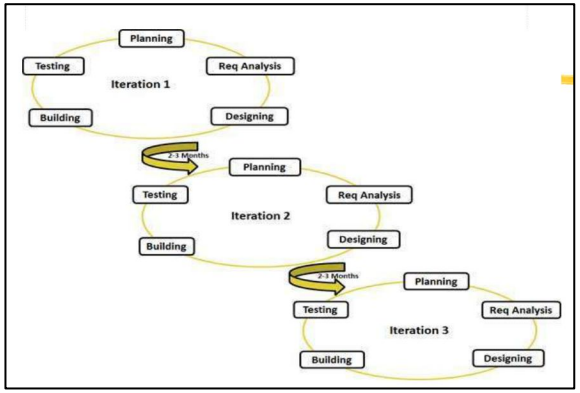
1. Classic Waterfall model
2. Iterative Waterfall model
3. Protype model
4. Evolutionary model
5. Spiral model
6. Rapid Application Development model
7. V-shaped model
8. Agile model

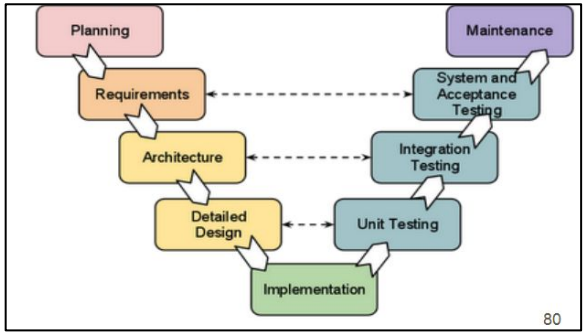

Feasibility Study
Req. Analysis
Design
Coding
Testing
Maintenance

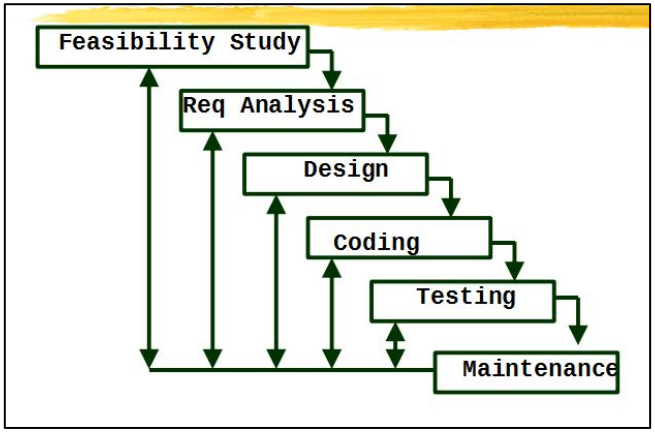

Team #1
Team #2
Team #3
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
60-90 days


A
A
B
C
A
B


Determine Objectives
Identify & Resolve Risks
Customer Evaluation of Prototype
Develop Next Level of Product


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Iteration 2
Iteration 3


Planning
Maintenance
Requirements
System and Acceptance Testing
Architecture
Integration Testing
Detailed Design
Unit Testing
Implementation
80


Feasibility Study
Req Analysis
Design
Coding
Testing
Maintenance