Software Engineering

Unit 1: Introduction to Software Engineering, Object-Oriented Methodology and Life Cycle

Introduction to Software Engineering, Object-Oriented Methodology and Life Cycle

1.1 Software Engineering:
- **1.1.1 Definition**
- **1.1.2 Program vs. Software**
- **1.1.3 Complexity of Software**
- **1.1.4 Characteristics of Software**

1.2 Object-Oriented Basic Concepts:
- **1.2.1 Classes and Object**
- **1.2.2 Messages and Attributes**
- **1.2.3 Encapsulation**
- **1.2.4 Inheritance**
- **1.2.5 Polymorphism**
- **1.2.6 Responsibility and Abstraction**
- **1.2.7 Object Composition**

1.3 Object-Oriented Methodologies:
- **1.3.1 Coad and Yourdon Methodology**

Introduction to Software Engineering, Object-Oriented Methodology and Life Cycle

- 1.3.2 Booch Methodology
- 1.3.3 Rumbaugh Methodology
- 1.3.4 Jacobson Methodology

1.4 Conventional Software Life Cycle Models:

- 1.4.1 Waterfall
- 1.4.2 Prototyping
- 1.4.3 Iterative Enhancement
- 1.4.4 Spiral Model

1.5 Agile Model:

- 1.5.1 Extreme Programming
- 1.5.2 Scrum

1.6 Object-Oriented Software Life Cycle Models:

- 1.6.1 Fountain Model
- 1.6.2 Rational Unified Process Model

CE: What is Software?

- The product that software professionals ***build*** and then ***support*** over the long term.

- For Examples:
  - Adobe Reader
  - Photoshop
  - Flash
  - Visual Studio
  - Netbeans
  - Railway Ticket Reservation System – IRCTC
  - 7-Zip

CE: Software Products

- Software Products are of two types:
  1. Generic Products
  2. Customized Products

1. Generic Products:

- Stand-alone systems that are marketed and sold to any customer who wishes to buy them.

- For Examples:
  - PC Software such as…
    Editing, Graphics Programs, Project Management Tools, etc.

  - Software for specific markets such as…
    Appointments systems for Dentists, etc.

CE: Software Products (Conti…)

2. Customized Products:

- Software that is commissioned (Custom-built / Ordered) by a specific customer to meet their own needs.

- For Examples:
  - Embedded Control Systems,
  - Air Traffic Control Software,
  - Traffic Monitoring Systems, etc.

CE: Why Software is important?

- The economies of ALL developed nations are dependent on software.

- More and more systems are software controlled, like….
  - Transportation,
  - Medical,
  - Telecommunications,
  - Military, Industrial,
  - Entertainment, etc.

- ***Software Engineering*** is concerned with *theories*, *methods* and *tools* for professional software development.

CE: 1.1
Software Engineering

1.1.1 Software Engineering

What is Engineering?

- Engineering forces us to focus on..
  - Systematic,
  - Scientific and
  - well-defined

  processes to produce a good quality product.

- Engineering activities believe…
  - to produce proper documents after every stage of software development.

- The development should be carried out as per the well-documented standards and guidelines.

1.1.1 Software Engineering (Conti…)

- It should be also completed within time and within budget.
- The same is expected in software development and all certifying agencies like SEI-CMM, ISO, IEEE, ANSI, etc.

SEI-CMM: Software Engineering Institute at Carnegie-Mellon University

- These all agencies are directing us to document all the processes and produce meaningful documentation after every stage of software development.

1.1.1 Software Engineering (Conti…)

- The main objective of the developer is…
  - to deliver good quality software to the customer and at low cost.

What is Software Engineering?

- The Software Engineering is a discipline that helps us …
  - to achieve this objective and
  - also increases the possibility of survival of the software during operation and maintenance.

1.1.2 Program vs. Software

What is Program?

- Program is a set of instructions written for a specific purpose.

- It may contain comment instructions (statements), to enhance the **readability** and **understandability** of a program.

What is Software?

- Software is a combination of….
  - Program(s),
  - Documentation and
  - Operating Procedure Manuals.

1.1.2 Program vs. Software (Conti…)

Documentation:

- There are many documents produced during software development. Some of the documents are:
  1. Software requirements and specification document
  2. Software design document
  3. Test plan document
  4. Test suite document
  5. Source code

- These documents are used not only during software development, but they are also used for maintaining the software.

1.1.2 Program vs. Software (Conti…)

Operating Procedure Manuals:

- Manuals are prepared to help customer to install, operate and maintain the software.
- Some of the manuals are:
  1. Installation Manual
  2. System Administration Manual
  3. Beginner's Guide Tutorial
  4. System Overview
  5. Reference Guide

- They are delivered along with the program(s) to the customer at the time of release.

1.1.3 Characteristics of Software

- **Some of the important characteristics of software are:**

1. Software is developed or ENGINEERED, but it is not manufactured.
2. Software does not wear out
3. Flexibility of Software
4. Reusability of Software

1.1.3 Characteristics of Software

2. Software does not wear out

- There is no concept of ‘aging’ in software.

- For Example:

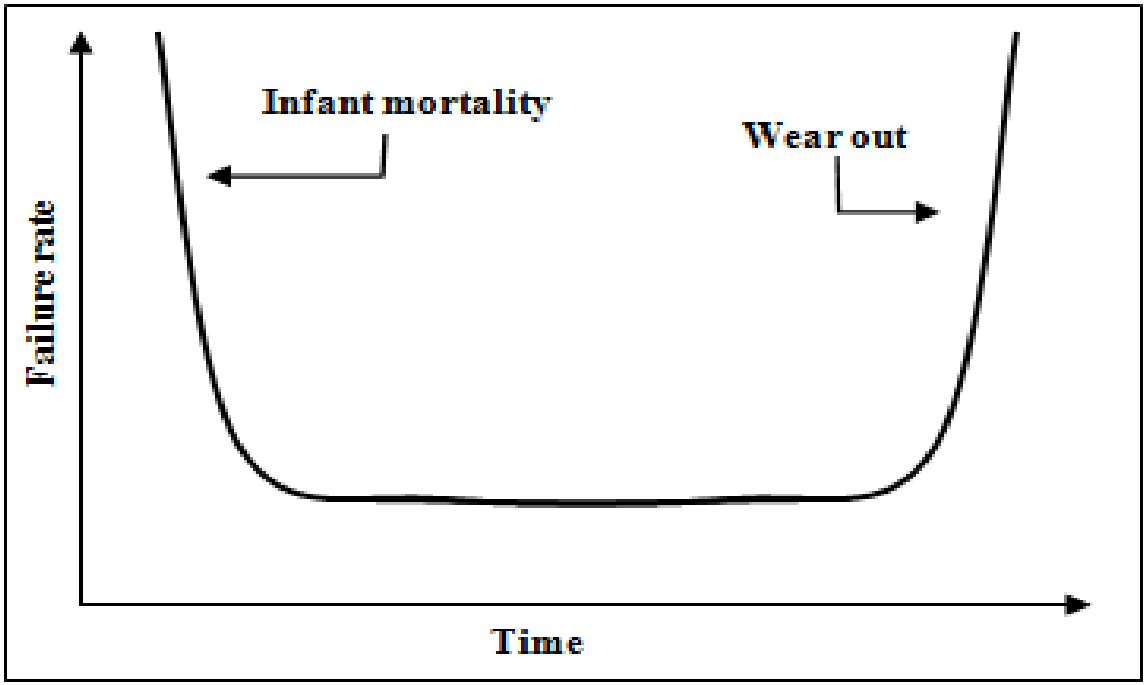


Figure 01: Failure curve for hardware

1.1.3 Characteristics of Software

2. Software does not wear out (Conti…)

- In the (burn-in) 1st phase, the failure rate is high. When we start testing the product, we detect faults.
- In the (useful life of any product) 2nd phase, the failure rate is more or less constant.
- In the (wear-out) 3rd phase, but after completion, of 2nd phase, the failure rate again increases, due to environmental conditions like temperature, humidity, pressure, etc.
  This means, various parts begin to wear-out.

But, we do not have the wear-out phase in software.

1.1.3 Characteristics of Software

2. Software does not wear out (Conti…)

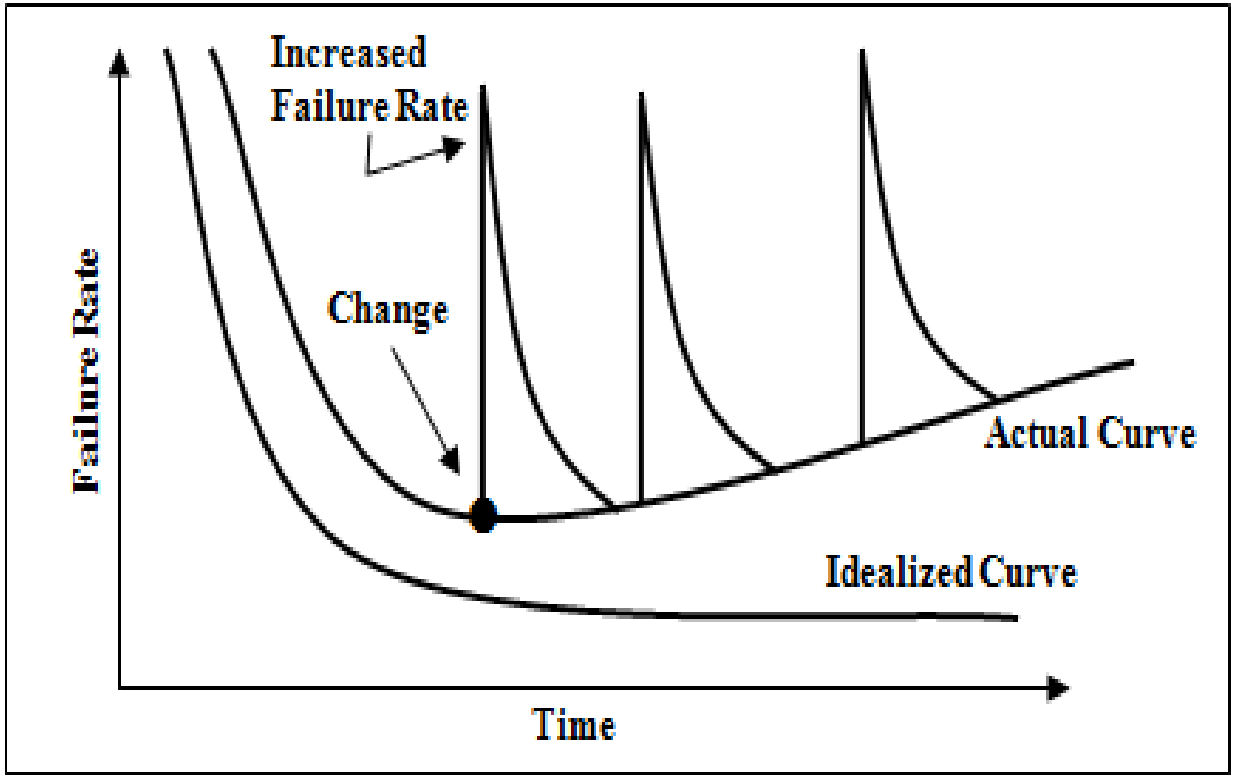


Figure 02: Failure Curves for software

1.1.3 Characteristics of Software

3. Flexibility of Software

- It may create confusion in stakeholders.
- They feel that, **any change can be possible at any time without much difficulty.**
- A stakeholder is a person who is having direct or indirect interest in the software system, including customer, software developers, managers, users, testers, etc.
- But, to modify source code, it is very difficult to manage.
- Addition of new functionality at later stages always makes the life difficult for software developers.

1.1.3 Characteristics of Software

3. Flexibility of Software (Conti…)

- Therefore, making changes in *software* is easier as compared to *hardware*.
- The software changes must be made after proper study and analysis of the source code.

1.1.3 Characteristics of Software

4. Reusability of Software

- The components of reusability is almost in all the disciplines of engineering.
- Suppose, we purchased monitor, keyboard, motherboard, microprocessor, switched-mode power supply(SMPS), etc. from the market.
- By using all these product, we integrate all the components as specified design and guidelines; and at last we test the product as per testing.
- Therefore, we have reuse those components for manufacturing of computers.

1.1.3 Characteristics of Software

4. Reusability of Software (Conti…)

- The reusable components are very common and popular in practice.
- It also reduces time and cost of manufacturing, by maintaining the quality standards.
- But, in software companies, most of the projects are for the specific customers; and every customer has unique requirements and expectations.
- The concept of reusability, is rarely used; and that has resulted into the development of costly software with more time.

1.1.3 Characteristics of Software

4. Reusability of Software (Conti…)

- The reusability has introduced an emerging area of study known as CBSE (components-based-software-engineering)

- Therefore, the object-oriented concepts may fulfil the long-standing desire of SE, which can be easily integrated in order to produce a good quality produce.

CE: 1.2
Object-Oriented Basic Concepts

CE 1.2 Object-Oriented Basic Concepts

Why is **object oriented** software development taking centre stage in software industry?

Why **object oriented** version of existing software products are coming in the market?

CE 1.2 Object-Oriented Basic Concepts (Conti…)

- We feel that real strength of **object oriented approach** is… its modeling ability to represent real world situations.
- A model helps us to ***visualize*** and ***understand*** a real situation along with its behavior.
- Architects/Designers use models to demonstrate their conceptual constructs which may also increase the confidence of their clients in the terms of ***design***, ***aesthetics*** and ***feel*** of the proposed project.

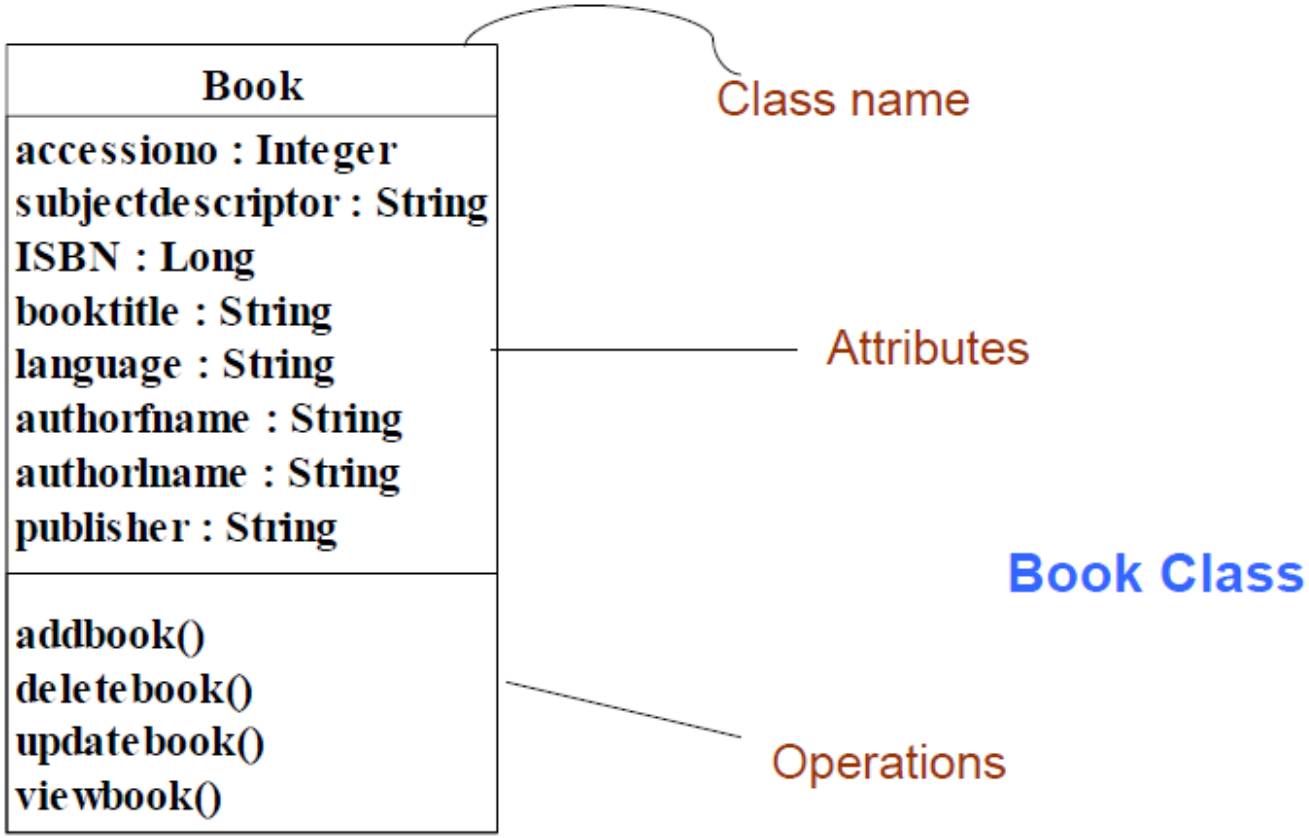
1.2 Object-Oriented Basic Concepts (Conti…)

- In Object-Oriented approach, real-world situations are effectively represented at any level of abstraction like…

1. Classes and Object
2. Messages, Attributes and Methods
3. Encapsulation
4. Inheritance
5. Polymorphism
6. Responsibility and Abstraction
7. Object Composition

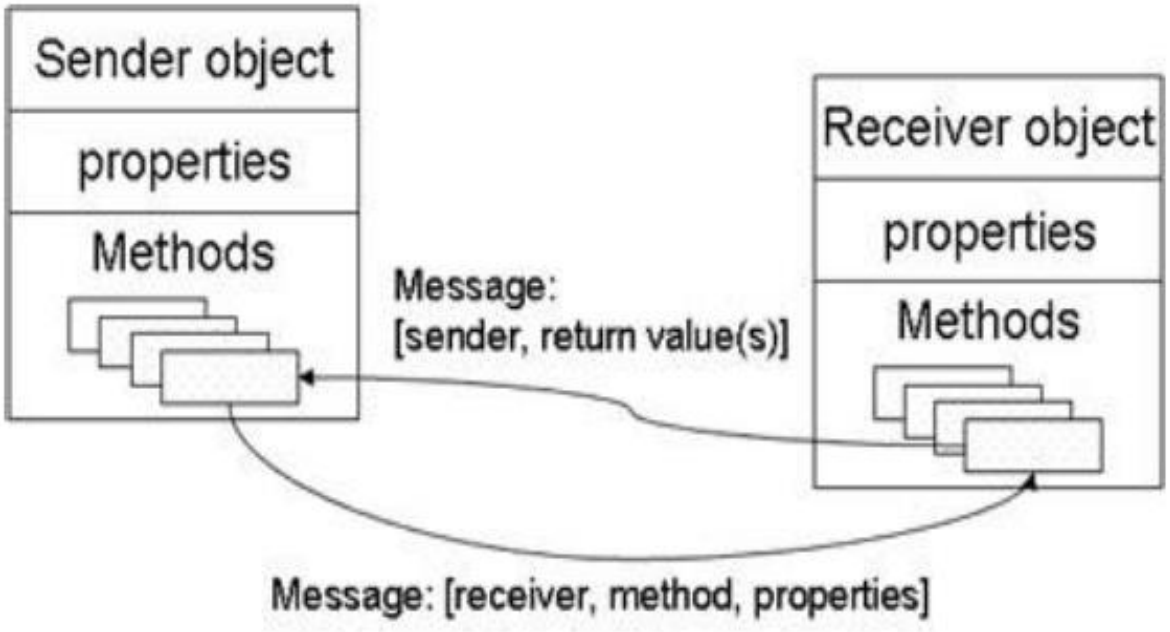
1.2 Object-Oriented Basic Concepts (Conti…)

1. Classes and Object:



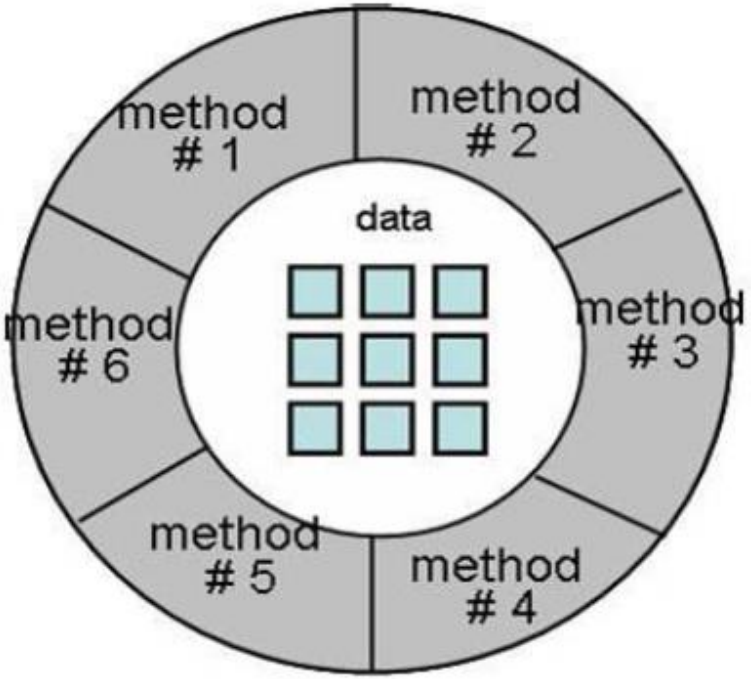
1.2 Object-Oriented Basic Concepts (Conti…)

2. Messages, Attributes and Methods:



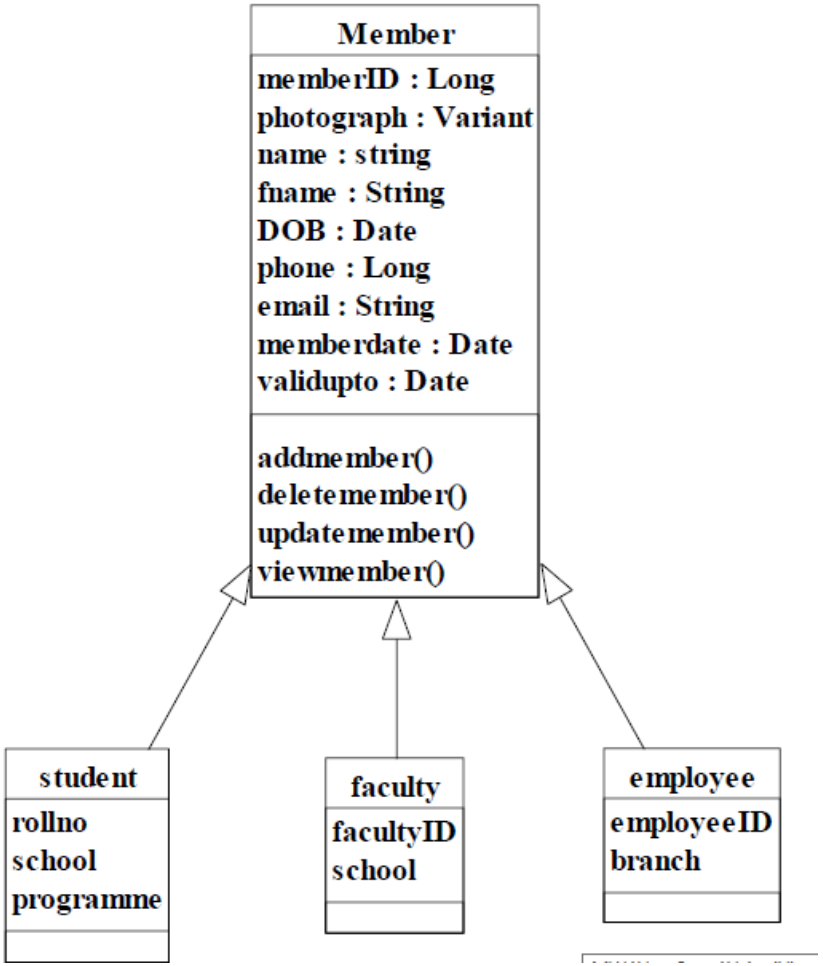
1.2 Object-Oriented Basic Concepts (Conti…)

3. Encapsulation:



1.2 Object-Oriented Basic Concepts (Conti…)

4. Inheritance:



1.2 Object-Oriented Basic Concepts (Conti…)

4. Inheritance: (Conti...)

```
class Member
{
    private:
       long int memberID;
       char photograph[300];
       char name[50];
       char fname[50];
       date DOB;
       long int phone;
       char email[60];
       date memberdate;
       date validupto;
  public:
          addmember();
          deletemember();
          updatemember();
          viewmember();
};
```

```
class employee: public Member //publiclyderivedclass
{
      long int employeeID;
      char branch[60];
};
class faculty: public Member //publicinheritance
{
      long int facultyID;
      char school[100];
};
class student: public Member //publicinheritance
{
      long int rollno;
      char school[100];
      char programme[100];
};
```

1.2 Object-Oriented Basic Concepts (Conti…)

5. Polymorphism

1.2 Object-Oriented Basic Concepts (Conti…)

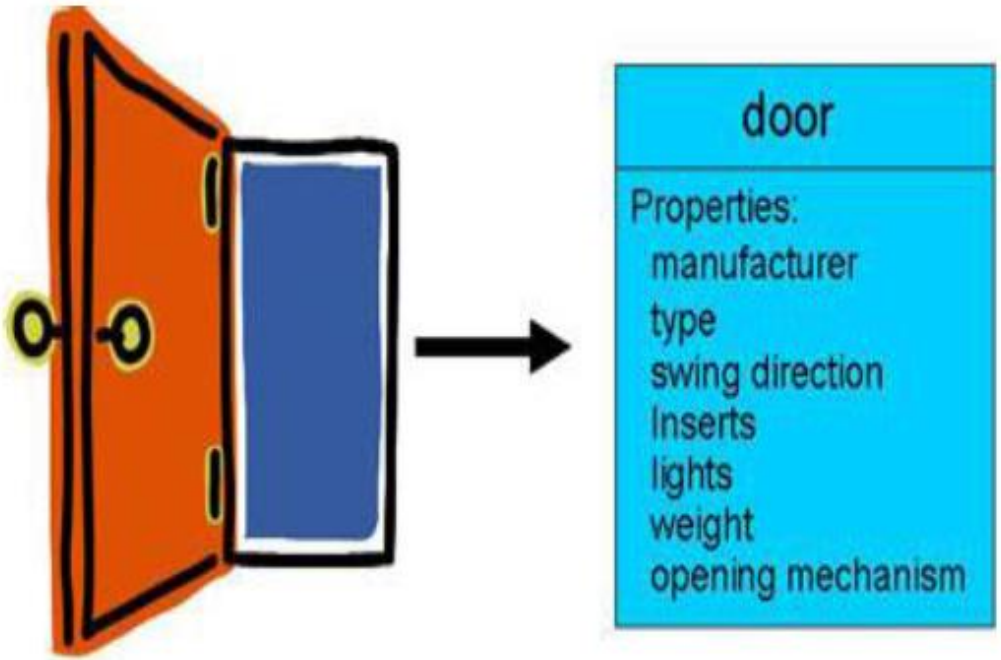
5. Polymorphism:

- The dictionary meaning of polymorphism is “many forms”.
- In the real world, the same operations may have different meanings in different situations.
- Same message is sent to different objects irrespective of their class, but the responses of objects may be different.

1.2 Object-Oriented Basic Concepts (Conti…)

6. Data Abstraction:

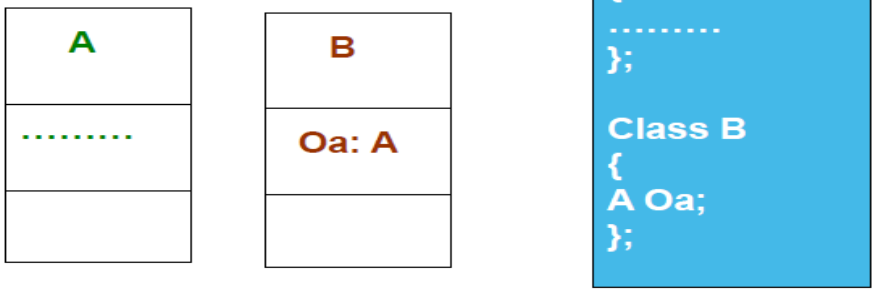
- Data abstraction is to collect essential elements combining to a complex data.



1.2 Object-Oriented Basic Concepts (Conti…)

7. Object Composition:

- The use of objects as data members in another class is referred to as object composition.
- The object is a collection of set of objects represented through has-a relationship.
- In object-oriented systems, has-a relationship shows that an object is declared as an attribute in another class.



CE: 1.3
Object-Oriented Methodologies

1.3 Object-Oriented Methodologies

- **Object-oriented is analyzed** by **Coad and Yourdon**.
- **Object-oriented is designed** by **Booch**.
- **Object modeling technique** by **Rumbaugh al**.
- **Object-oriented software engineering** by **Jacobson.**

1.3 Object-Oriented Methodologies

1. Coad and Yourdon Methodology:

- Identification of classes and objects:
  - ✓ involves in studying the application domain (field) and the system's environment.
  - ✓ The behavior of each objects are found and this information is documented.

- Identification of structures:
  - ✓ involve identification of is-a and whole-part relationships.
  - ✓ The ***is-a relationship*** captures class inheritance (known as Gen-Spec structure) and
  - ✓ The ***whole-part relationship*** captures the information that how an object is part of another object.

1.3 Object-Oriented Methodologies

1. Coad and Yourdon Methodology: (Conti…)

- Definition of subjects:
  - ✓ Each structure is classified into a subject.

- Definition of attributes:
  - ✓ Attributes are the data members of the class.
  - ✓ The attributes for each object are defined and kept at the appropriate level in the inheritance hierarchy.

- Definition of services (methods):
  - ✓ involve identification of operations in a class.
  - ✓ This also involves identification of interfaces amongst the objects through messages.

1.3 Object-Oriented Methodologies

2. Booch Methodology:

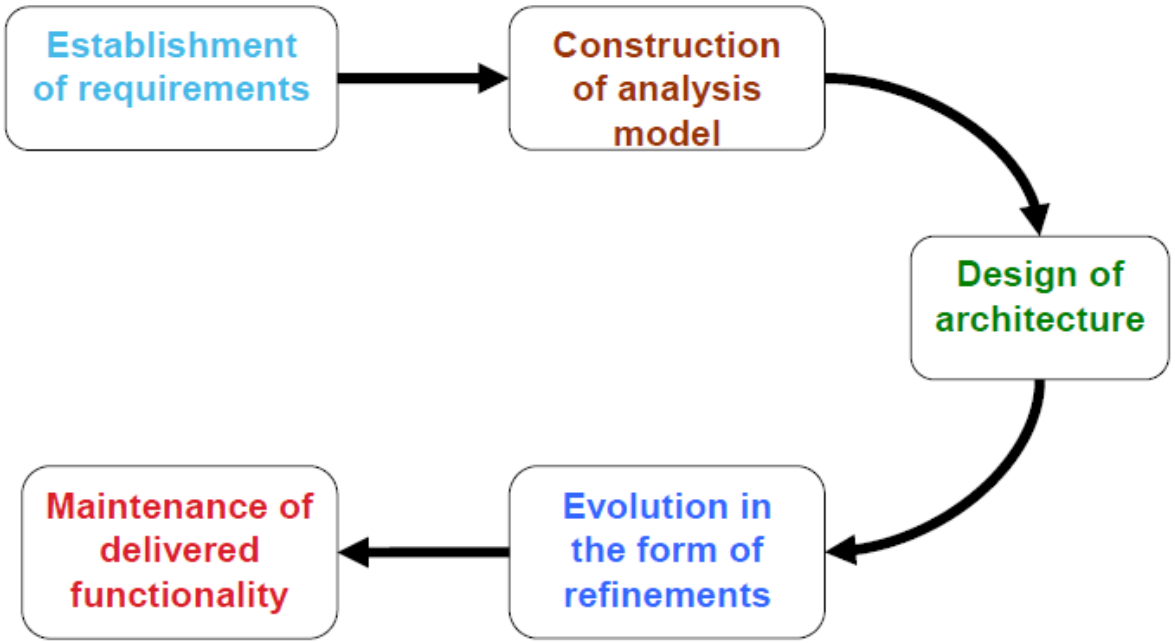
- Grady Booch proposed **object oriented methodology** in his book i.e. Object-Oriented Design (OOD) in 1991.

- The primary aim of OOD was to establish a base for implementation of object oriented systems.

- The Booch methodology can be broadly divided into two processes:
    1. Macro Process
    2. Micro Process

1.3 Object-Oriented Methodologies

2. Booch Methodology: (Conti…)



1.3 Object-Oriented Methodologies

2. Booch Methodology: (Conti…)

- In first phase the requirements are established using context diagrams and prototypes.
- The outcomes of this phase are core requirements of the system.
- ***Analysis process*** involves requirement capturing and understanding. It involves "what of the system".
  - ✓ This phase consists of construction of use cases, identification and prioritization of risks.

1.3 Object-Oriented Methodologies

2. Booch Methodology: (Conti…)

- ***Design phase*** focuses on construction of architecture of the system and involves:
  - Identification of horizontal layers
  - Mapping classes to subsystems
  - Release planning
  - Attaching risks identified in analysis phase to releases.

- ***Evolutionary phase*** involves implementation of the system and each release adds to the functionality of the system.

- ***Maintenance phase*** consists of post deployment activities.

1.3 Object-Oriented Methodologies

2. Booch Methodology: (Conti…)

- It is the lower level process. The following recursive steps are followed in OOD micro process
  - ✓ Identification of classes and objects
  - ✓ Identification of semantics of classes and objects
  - ✓ Identification of relationship amongst classes and objects
  - ✓ Specification of interfaces and implementation of classes and objects

1.3 Object-Oriented Methodologies

3. Rumbaugh Methodology:

- Rumbaugh developed a technique that focuses on analysis, design and implementation of the system. This technique is popularly known as Object Technique (OMT).

- The OMT consists of four phases:
    1. Analysis
    2. System Design
    3. Object Design
    4. Implementation

1.3 Object-Oriented Methodologies

3. Rumbaugh Methodology: (Conti…)

1. Analysis Phase:

- Analysis phase is composed of three submodels given below:

    1. Object Model:
        - It captures the static aspect of the system.

    2. Dynamic Model:
        - It captures the behavioral aspects of the object models and describes state of the objects.

    3. Functional Model:
        - It represents the functional aspects of the system in terms of operations defined in the classes.

1.3 Object-Oriented Methodologies

3. Rumbaugh Methodology: (Conti…)

2. System Design Phase:

- In this phase high level design is developed taking the implementation environment including DBMS and communication protocols into account.

3. Object Design Phase:

- The goal of this phase is to define the objects in details.
- The algorithms and operations of the objects are defined in this phase. New objects may be identified to represent the intermediate functionality.

1.3 Object-Oriented Methodologies

3. Rumbaugh Methodology: (Conti…)

4. Implementation Phase:

- Finally the objects are implemented following coding standards and guidelines.

1.3 Object-Oriented Methodologies

4. Jacobson Methodology:

- All the methodologies described above still lack of a comprehensive architecture to develop a software project.

- The Jacobson's methodology known as "Object Oriented Software Engineering (OOSE)" consists of five models:

1. Requirement Model:
- The aim of the model is to gather software requirements.

2. Analysis Model:
- The goal of this model is to produce ideal, robust and modifiable structure of an object.

1.3 Object-Oriented Methodologies

4. Jacobson Methodology: (Conti…)

3. Design Model:

- It refines the objects keeping the implementation environment in mind.

4. Implementation Model:

- It implements the objects.

5. Test Model:

- The goal of the test model is to validate and verify the functionality of the system.

CE: 1.4
Conventional Software Life Cycle Models

1.4 Conventional Software Life Cycle Models

- The Conventional Software Life Cycle Models are of different types:
  1. Waterfall
  2. Prototyping
  3. Iterative Enhancement
  4. Spiral Model

1. Waterfall Model:

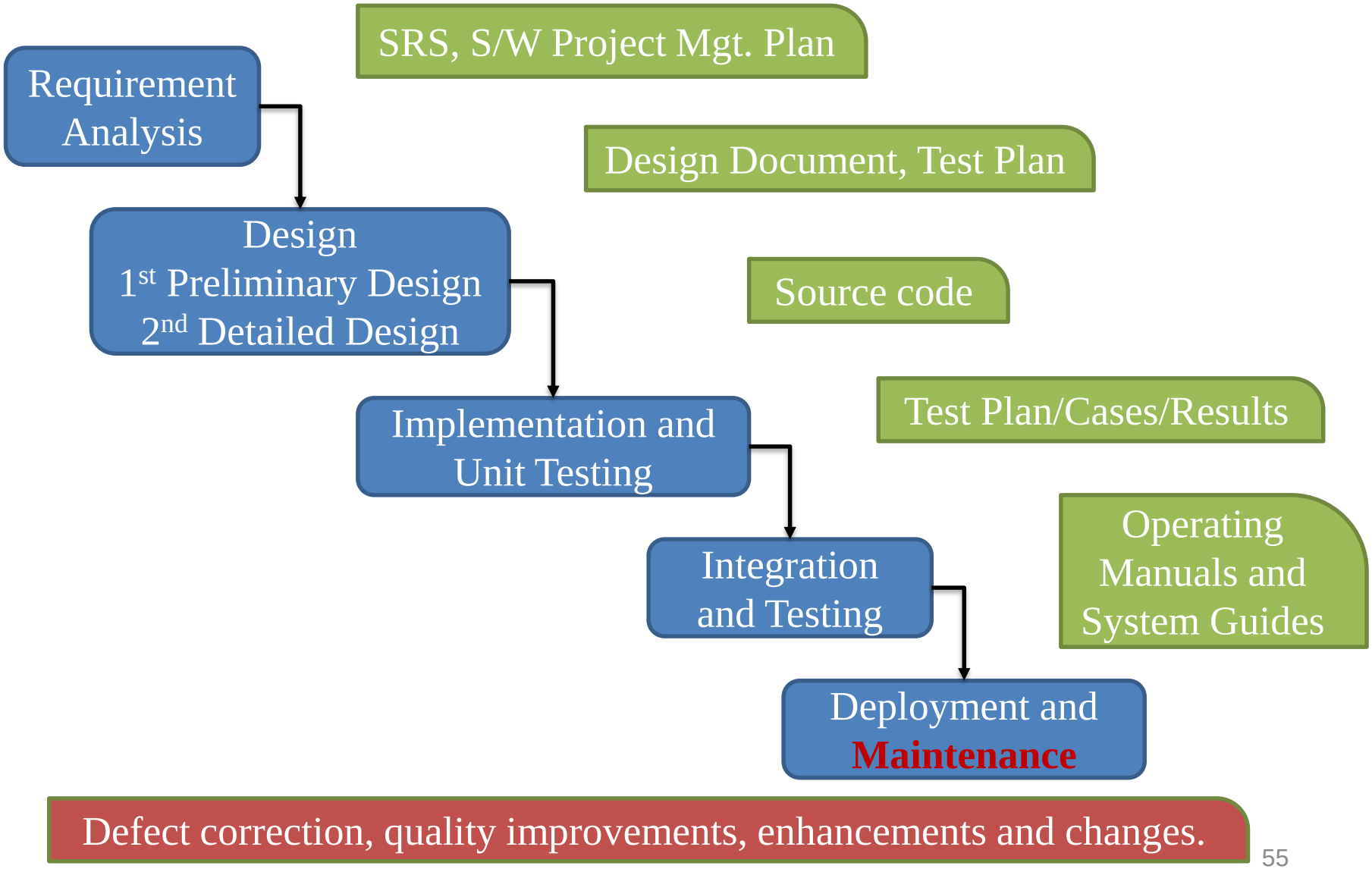
- **It has five phases:**
  1. Requirement Analysis
  2. Design
  3. Implementation and unit testing
  4. Integration and testing
  5. Deployment and maintenance

- The phases are completed in sequential order and do not overlap with each other.

- The phase should be completed before the commencement of the next phase.

- The output from one phase goes as input into the next phase.

1. Waterfall Model: (Conti…)



1. Waterfall Model: (Conti…)

- **Advantages:**

  - ✓ Simple and easy to understand and use
  - ✓ Understandability and different phase with their own set of functions.
  - ✓ It is highly suitable for the projects, because the requirements are completely known in the beginning of the project.

1. Waterfall Model: (Conti…)

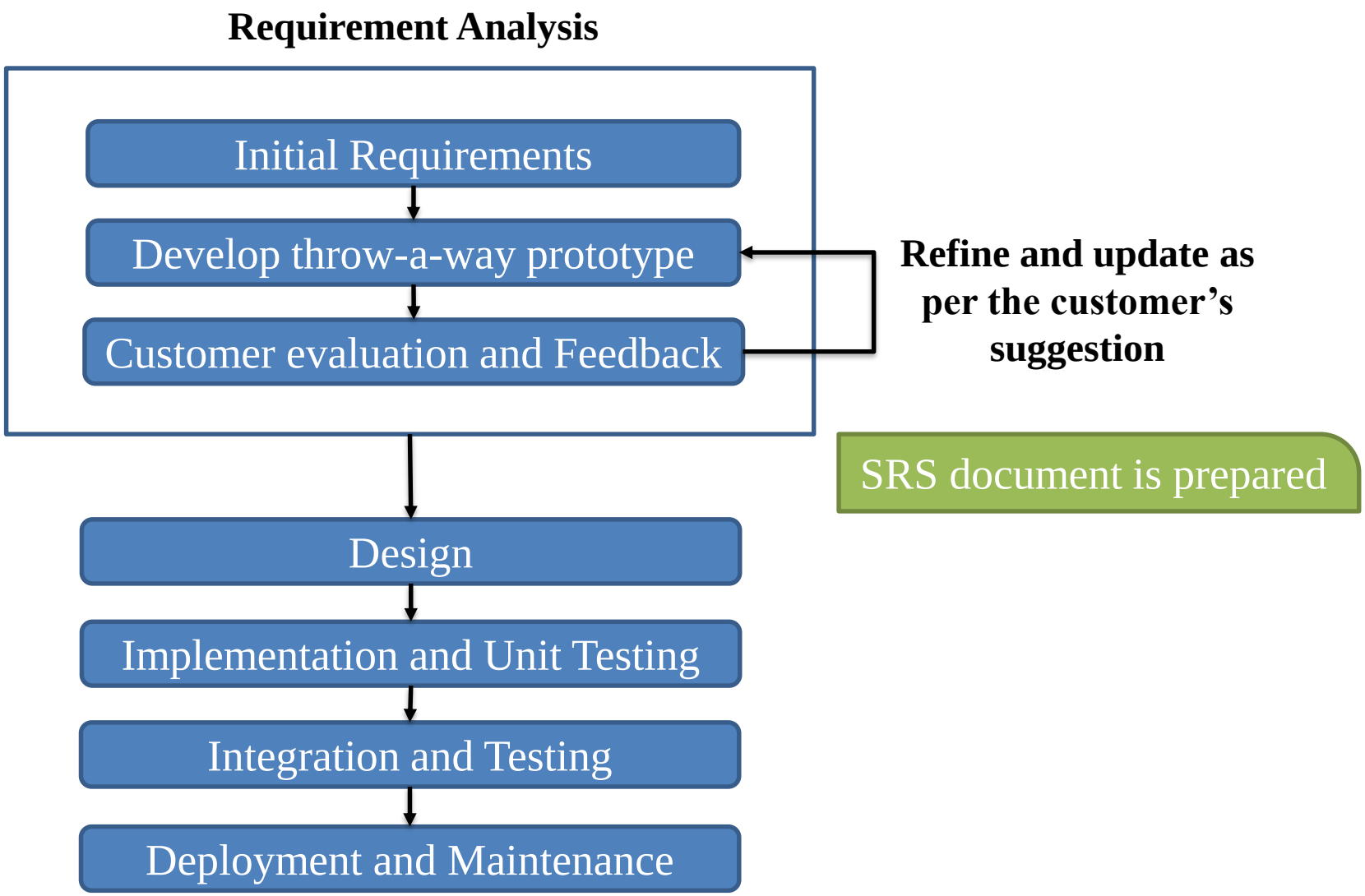
- **Disadvantages:**

1. Large number of documents are produced.
2. No working software is produced until late during the life cycle.
3. High amounts of risk and doubt.
4. Not a good model for complex and object-oriented projects.
5. Poor model for long and ongoing projects.
6. Not suitable for the projects where requirements are at a reasonable to high risk of changing. So, risk and uncertainty is high with this process model.
7. It is difficult to measure progress within stages.
8. Cannot accommodate changing requirements.
9. Adjusting scope during the life cycle can end a project.
10. Integration is done as a “big-bang” at very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

2. Prototyping Model:

- This model is constructed in order to determine and understand the customer's need.
- The customer evaluates this prototype, and then the final SRS document is prepared on the basis of refined requirements.
- The prototype is generally built quickly to give the customer, the feel of the system.

2. Prototyping Model: (Conti…)



2. Prototyping Model: (Conti…)

- **Advantages:**

1. Stable requirements:
   - Stabilized by refining the prototype again and again based on the customer's feedback.

2. High-Quality system:
   - Quality of the system will be high, because developers and customers have gained a lot of experience.

3. Low cost:
   - The actual cost of the system will be reduced.

2. Prototyping Model: (Conti…)

- **Disadvantages:**

1. The customer may expect quick delivery of the software.

2. The time for development may exceed in some cases.

2. Prototyping Model: (Conti…)

- But, the prototype will finally be discarded, because it need not be flexible, maintainable and fault tolerant.

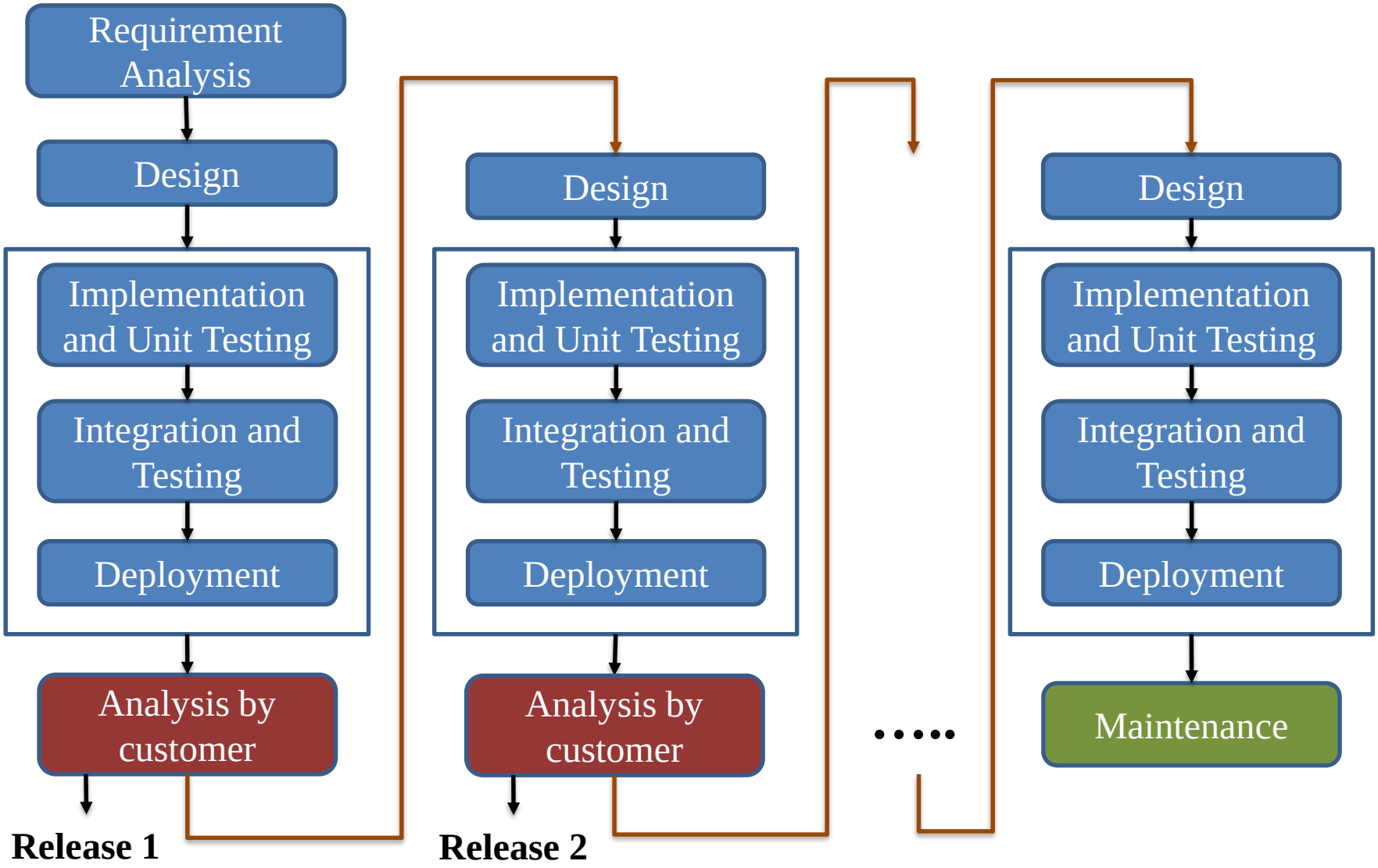
3. Iterative Enhancement Model:

- It is also known as Incremental Model.
- The problem of waterfall model is overcome in the Iterative Enhancement Model.
- It combines the advantages of both the waterfall model and prototype model.
- This model is same as in the waterfall model, but it is carried out in many cycles.
- A usable product is delivered at the end of each cycle. This allows to obtain the customer's feedback after each cycle.

3. Iterative Enhancement Model: (Conti…)

- Therefore, each cycle provides an additional functionality.
- As the product is developed in increments, the testing of the product also becomes easy.
- A project control list is created, which consist of a set of tasks to be completed, before the final delivery of the product.
  (This list also keeps the track of the required work that is needed to be completed before the system is ready for final deployment.)
- The Iterative Enhancement Model broadly consists of three iterative phases: ***Design, Implementation*** and ***Analysis by customer***.
- After the ***Requirement Analysis***, these 3 phases are carried out repeatedly until complete functionality is delivered to the customer.

3. Iterative Enhancement Model: (Conti…)



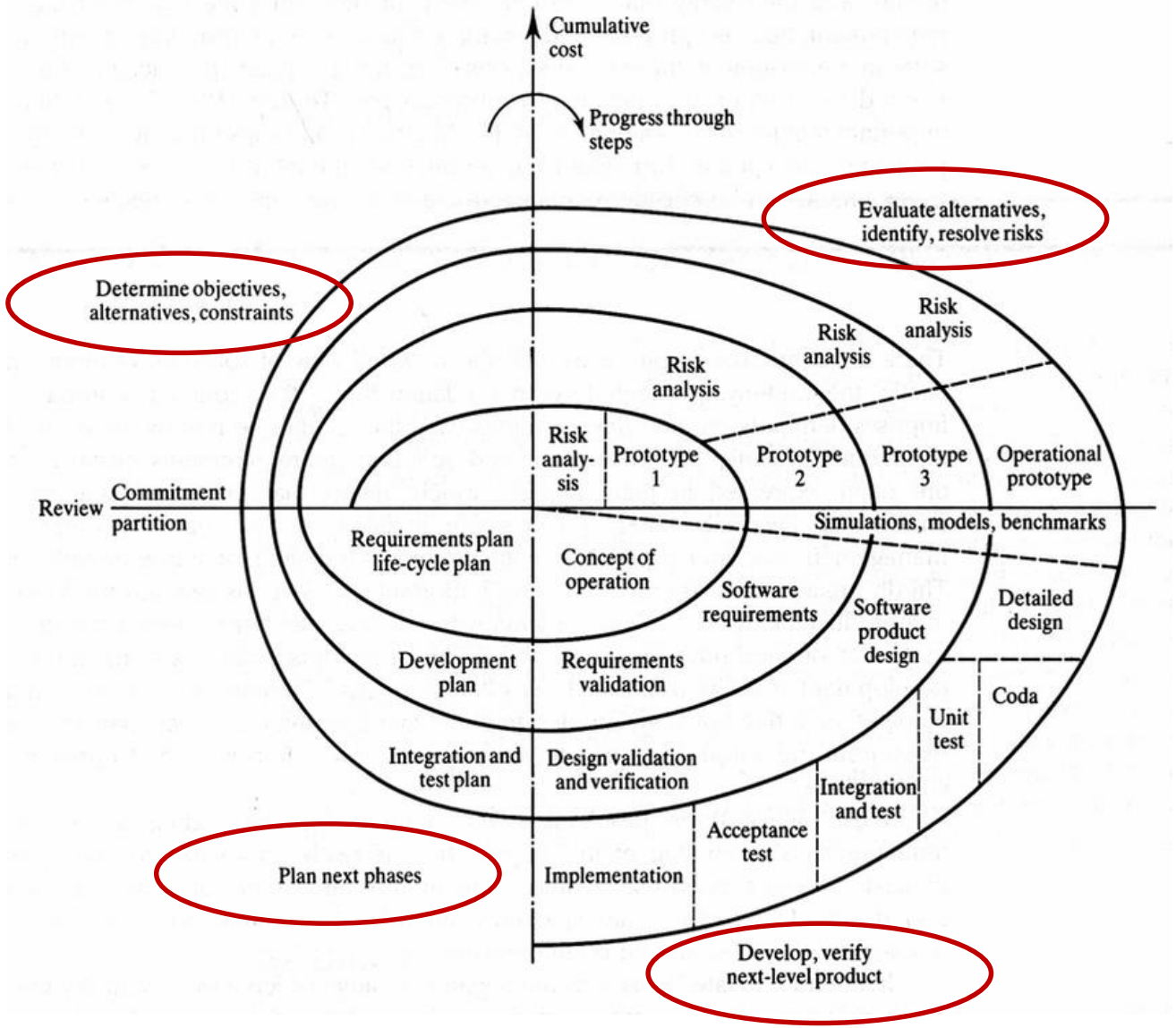
3. Iterative Enhancement Model: (Conti…)

- Therefore, with the help of various cycles, the customer can see the progress of the software continuously.
- It may also increase the cost, because the later software increments may require modifications.

4. Spiral Model:

- A Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling.
- In its diagrammatic representation, it looks like a spiral with many loops.
- The Spiral model is developed by Barry W. Boehm in 1986.
- The Spiral model is divided into four phases:
  1. Determine objectives, alternatives and constraints.
  2. Evaluate alternatives, identify and resolve risks.
  3. Develop and verify next-level product.
  4. Plan next phases.

4. Spiral Model: (Conti…)



4. Spiral Model: (Conti…)

1. Determine objectives, alternatives and constraints:

- Each cycle of spiral model begins with identification of objectives (functional and non-functional requirements),
- Determination of alternatives means implementing the specified part of the product like design 1, design 2, reuse, buy, etc.
- Identification of constraints imposed on the completion of the portion of the product (resources, cost, schedule, etc.)

2. Evaluate alternatives, identify and resolve risks.

- Its means to identify the significant project risks and resolve that risk.
- This may apply risk resolution techniques like prototyping, customer interviews, modelling, etc.

4. Spiral Model: (Conti…)

3. Develop and verify next-level product:

- It means addressing all the major risk-related issues.
- When all the performance, development and interface-related risks are resolved, the waterfall approach is followed incrementally in the next steps.

4. Plan next phases:

- Each level incorporates validation of the activities by conducting reviews and preparing various plans for the next cycle.

4. Spiral Model: (Conti…)

- The spiral model repeats the four phases rounds:

Round 0: Feasibility study

- To determine the feasibility of the system in terms of…
  - ✓ Resources,
  - ✓ Cost,
  - ✓ Schedule and
  - ✓ Productivity.

Round 1: Concept of operation

- The output of this round is a basic requirement of the life cycle plan for the implementation of the concept.

Round 2: Top-level requirements analysis

- Requirements are specified and validated for risk-driven approach.
- And the output produced is the development plan.

4. Spiral Model: (Conti…)

Round 3: Software design

- A primary design is created, verified and validated.
- Integration (Addition) and test plan is produced after the completion of this round.

Round 4: Design, implementation and testing

- Constructs the detailed design of the product, which includes coding, unit testing, integration testing and acceptance testing.

4. Spiral Model: (Conti…)

Disadvantages:

- Lack of expertise to determine and resolve risks.
- It is applicable only to large-sized projects.
- Sometimes the cost of performing risk analysis may overweigh.

CE: 1.5
Agile Model

CE: 1.5 Agile Model

What is Agile?

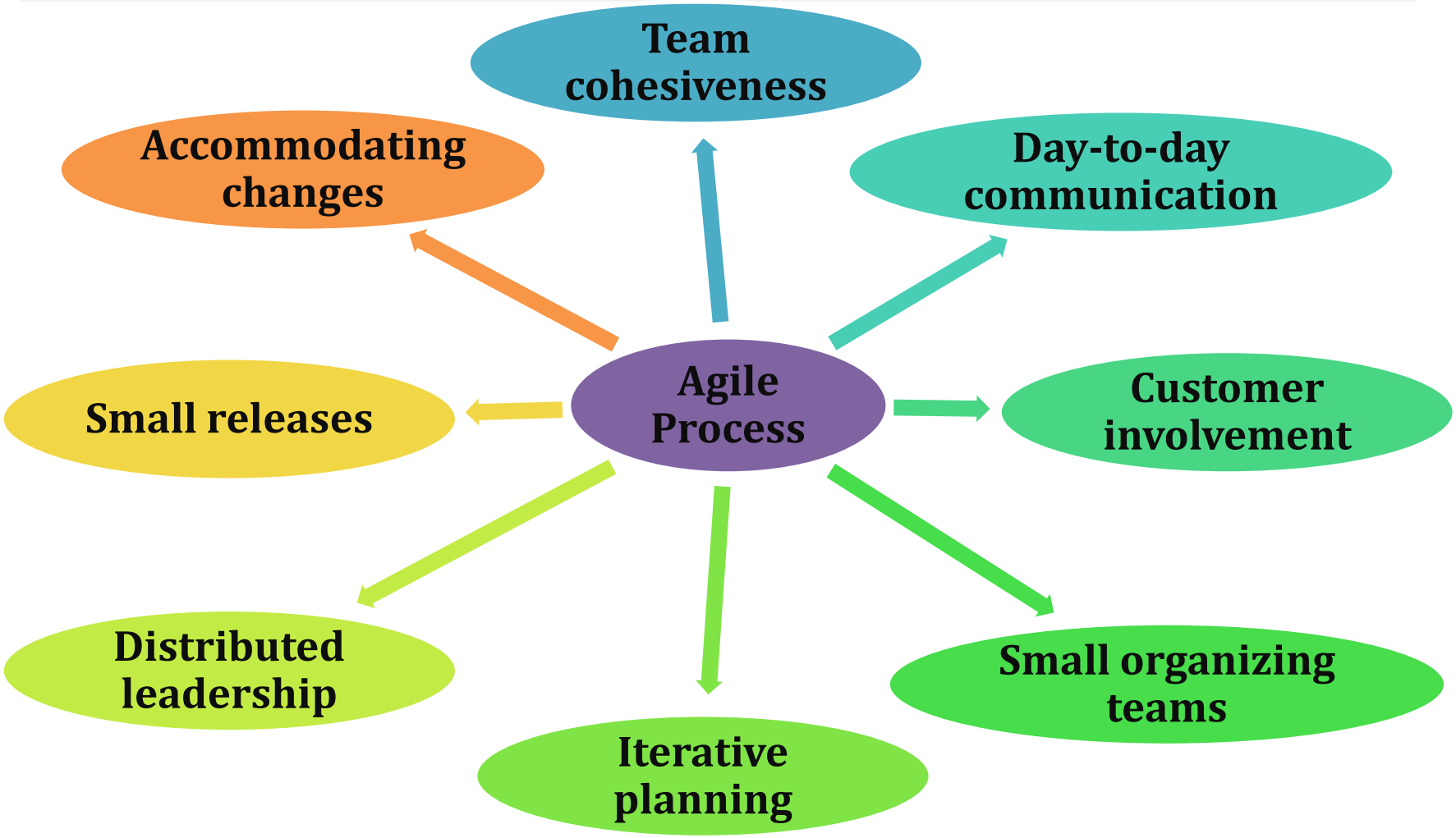
- Agile means the ability to move quickly and easily.

What is Agile process?

- The agile process is both quick and easy.

- Agile processes are based on team …..
  - ✓ cohesiveness,
  - ✓ quick feedbacks,
  - ✓ incremental development,
  - ✓ experienced developers and
  - ✓ automated testing.

CE: 1.5 Agile Model (Conti…)



Key features of agile processes.

CE: 1.5.1 Extreme Programming

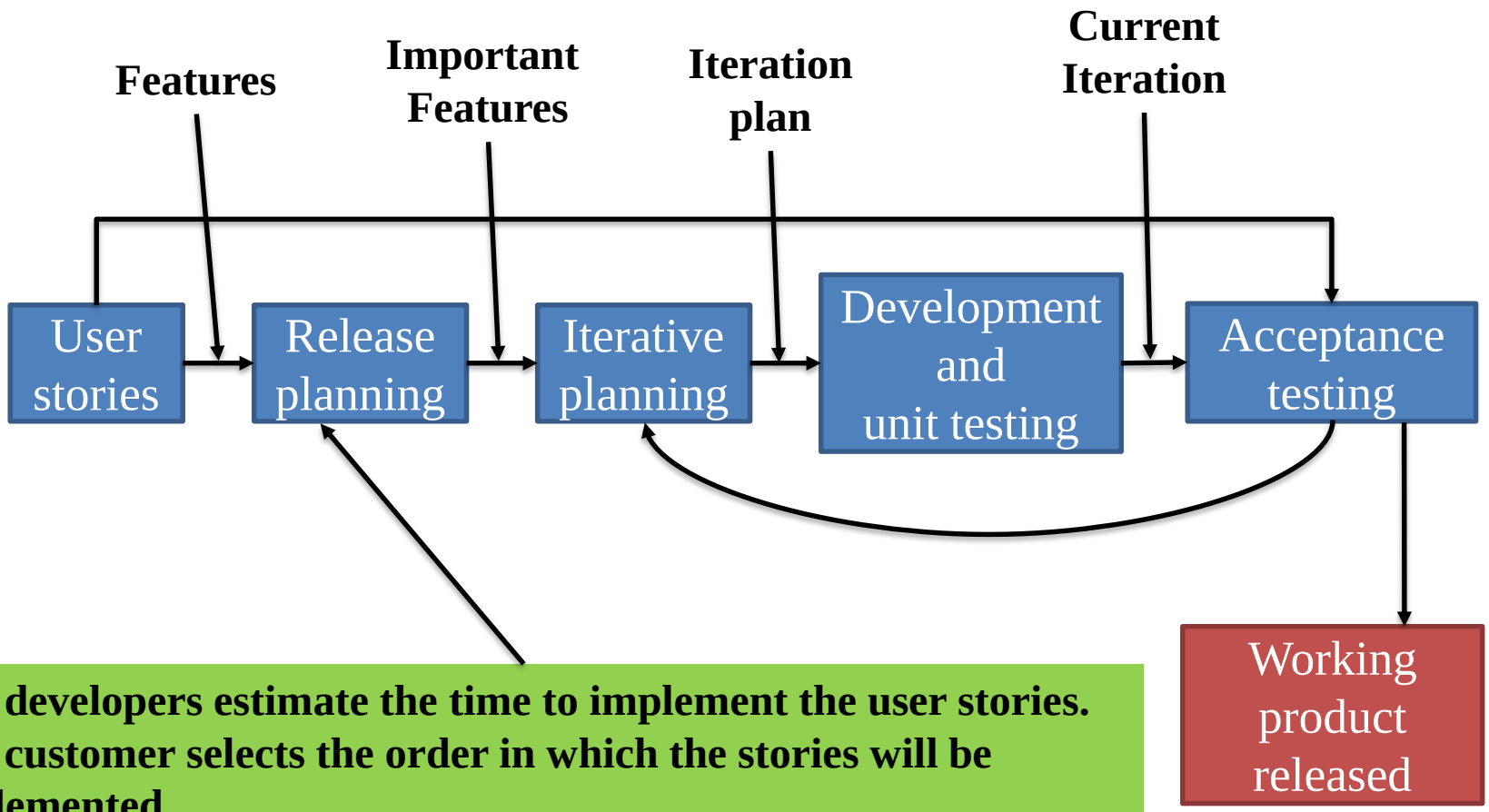
- Extreme Programming (XP) project started in the middle of the 1990s.
- It is based on the principles of agile processes.
- Therefore, XP follow the methodology of agile processes and focuses on customer satisfaction.
- So that, it emphasizes on continuous customer involvement and testing.

CE: 1.5.1 Extreme Programming (Conti…)

- It features includes:
  - ✓ Simple processes,
  - ✓ Continuous feedback,
  - ✓ Incremental growth,
  - ✓ High communication and
  - ✓ Courage.

CE: 1.5.1 Extreme Programming (Conti…)

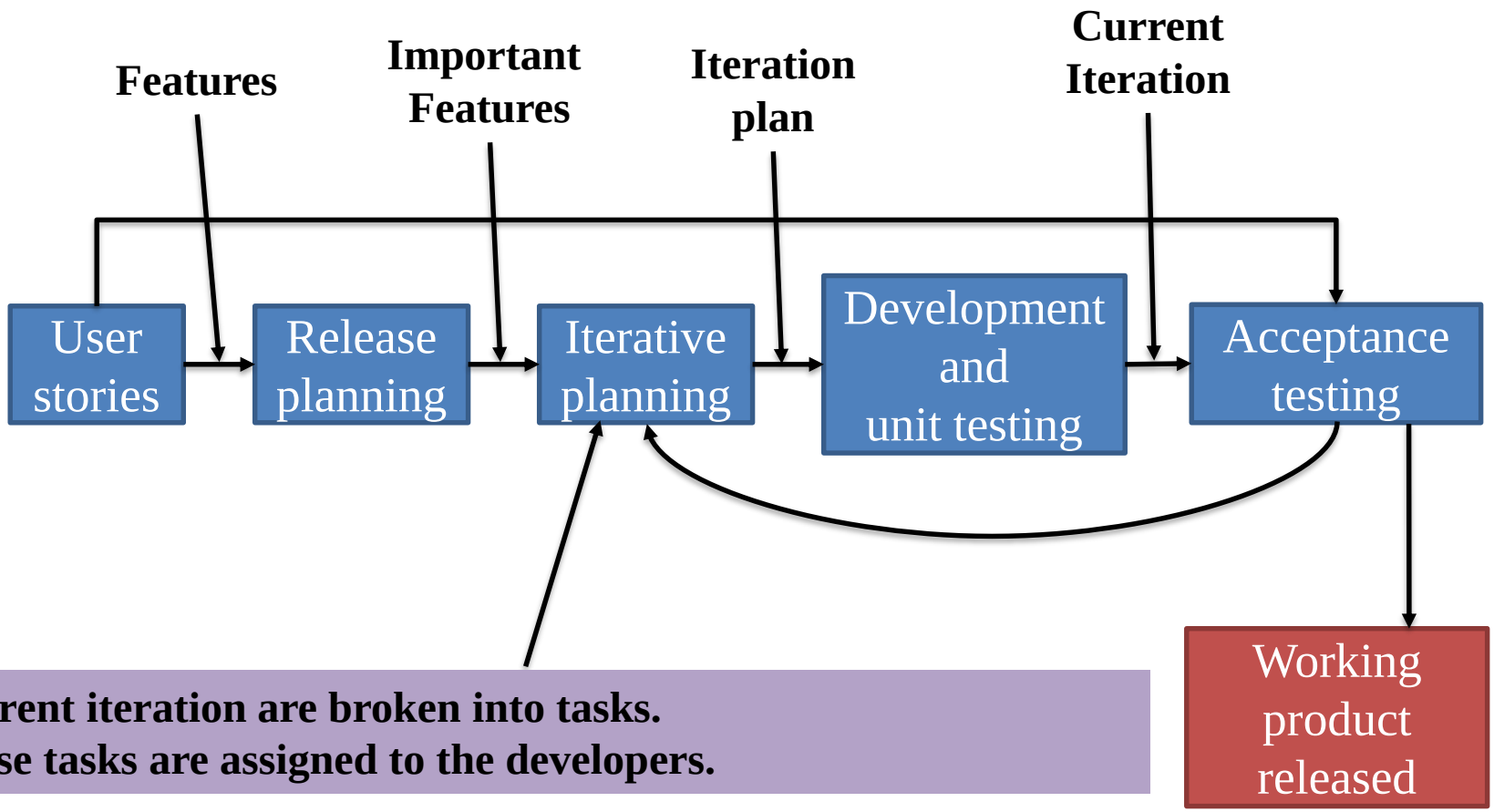
- **The typical life cycle of an XP**



The developers estimate the time to implement the user stories. The customer selects the order in which the stories will be implemented.

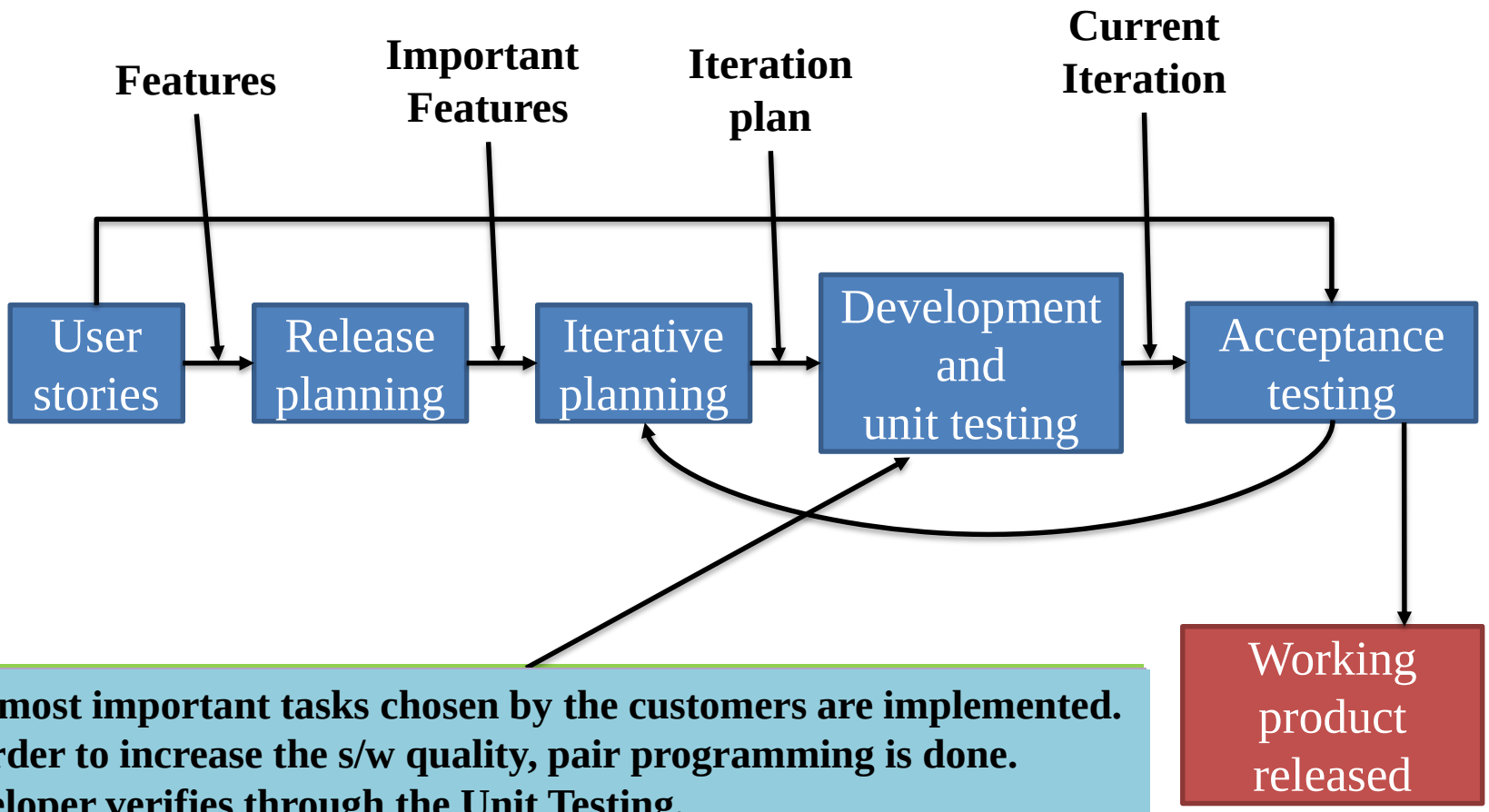
CE: 1.5.1 Extreme Programming (Conti…)

- **The typical life cycle of an XP**



CE: 1.5.1 Extreme Programming (Conti…)

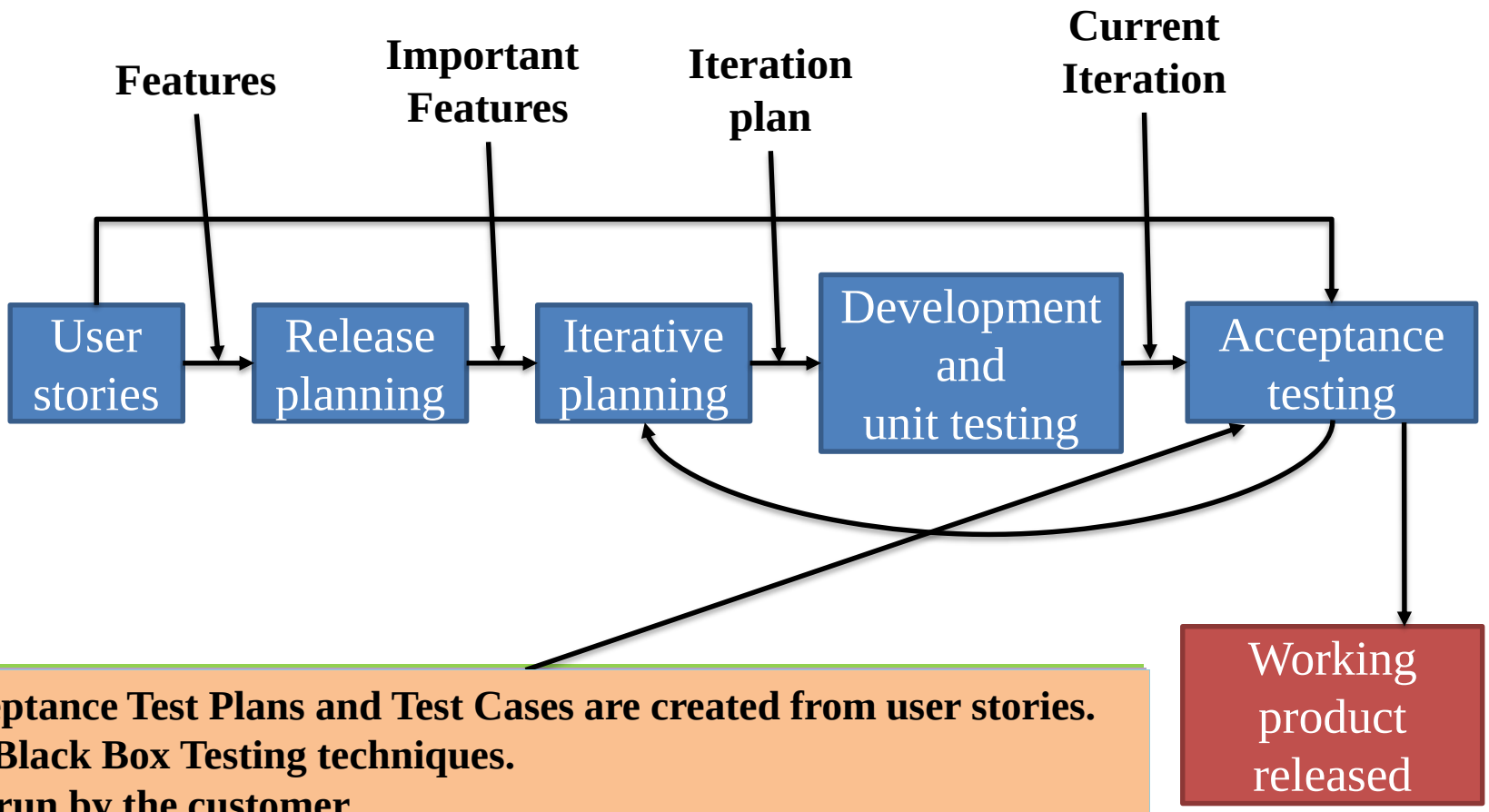
- **The typical life cycle of an XP**



The most important tasks chosen by the customers are implemented. In order to increase the s/w quality, pair programming is done. Developer verifies through the Unit Testing.

CE: 1.5.1 Extreme Programming (Conti…)

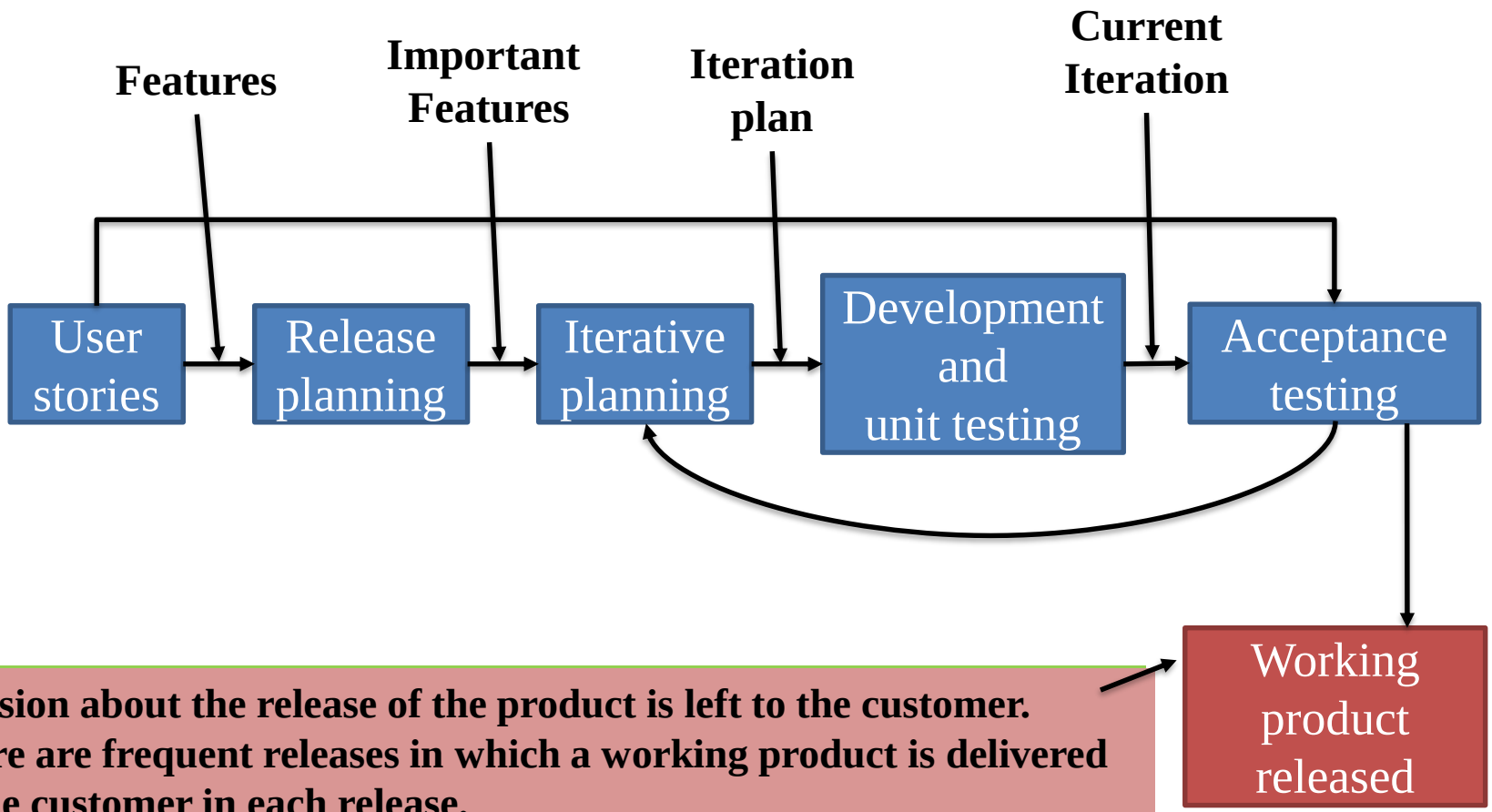
- **The typical life cycle of an XP**



Acceptance Test Plans and Test Cases are created from user stories.
Use Black Box Testing techniques.
It is run by the customer.

CE: 1.5.1 Extreme Programming (Conti…)

- **The typical life cycle of an XP**



Decision about the release of the product is left to the customer. There are frequent releases in which a working product is delivered to the customer in each release.

CE: 1.5.2 Scrum Model

- In the Scrum Model, a project is divided into small parts of work that can be incrementally developed and delivered over time boxes that are called ***sprints***.
- Therefore, the software gets developed over a series of manageable chunks.
- Each ***sprint*** typically takes only a couple of weeks to complete.
- At the end of each *sprint*, the ***stakeholders*** and ***team members*** meet to evaluate the progress made and the stakeholders suggest to the development team for the changes needed to features that have already been developed and any overall improvements that they might feel necessary.

CE: 1.5.2 Scrum Model (Conti…)

- In the Scrum Model, the team members assume three fundamental roles:
  1. Software Owner – is responsible for communicating the customers vision of the software to the development team.

  2. Scrum Master - acts as a link between the *software owner* and the *team member*, thereby facilitating the development work.

  3. Team Member

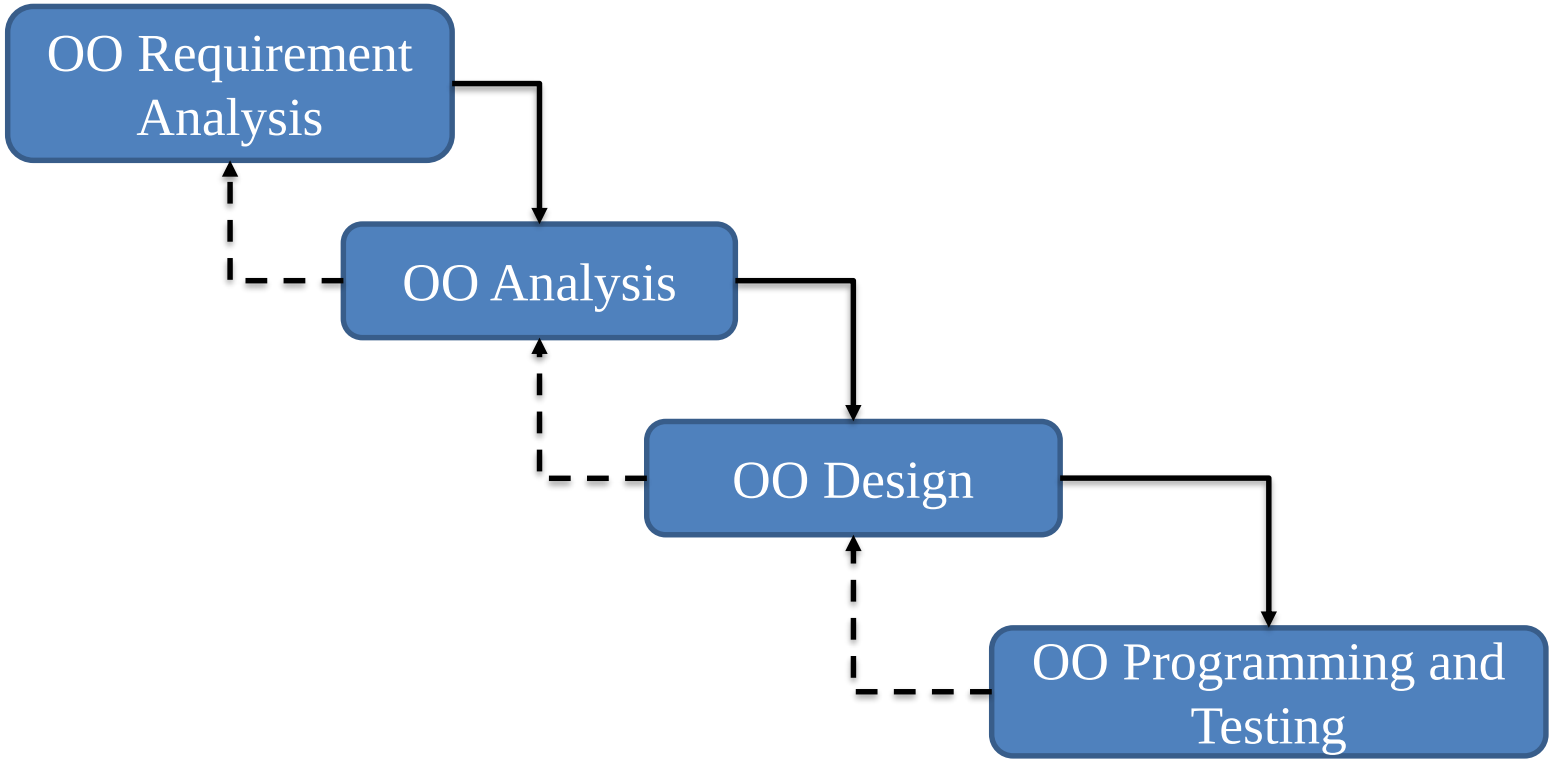
CE: 1.6
Object-Oriented Software Life Cycle Models

CE: 1.6 Object-Oriented Software Life Cycle Models

- Object-Oriented Software Development Life Cycle includes:
  1. Object-Oriented Requirement Analysis
  2. Object-Oriented Analysis
  3. Object-Oriented Design
  4. Object-Oriented Programming and Testing

CE: 1.6 Object-Oriented Software Life Cycle Models (Conti…)

- A typical Object-Oriented Life Cycle Model is represented as:



CE: 1.6 Object-Oriented Software Life Cycle Models (Conti…)

- Comparison between phases of traditional and object-oriented approaches:

	Traditional approaches	Object-oriented approaches
Methodology	Functional and process driven.	Object driven.
Requirement analysis phase	DFDs, Data Dictionary, ER diagrams	Use case for requirement capturing.
Analysis phase	DFDs, Data Dictionary, ER diagrams	Object identification and description, attributes and functions that operate on those attributes are determined.
Design phase	Structured chart, Flow chart, Pseudo code.	Class diagram, Object diagrams, Sequence diagrams, Collaboration diagrams.

CE: 1.6 Object-Oriented Software Life Cycle Models (Conti…)

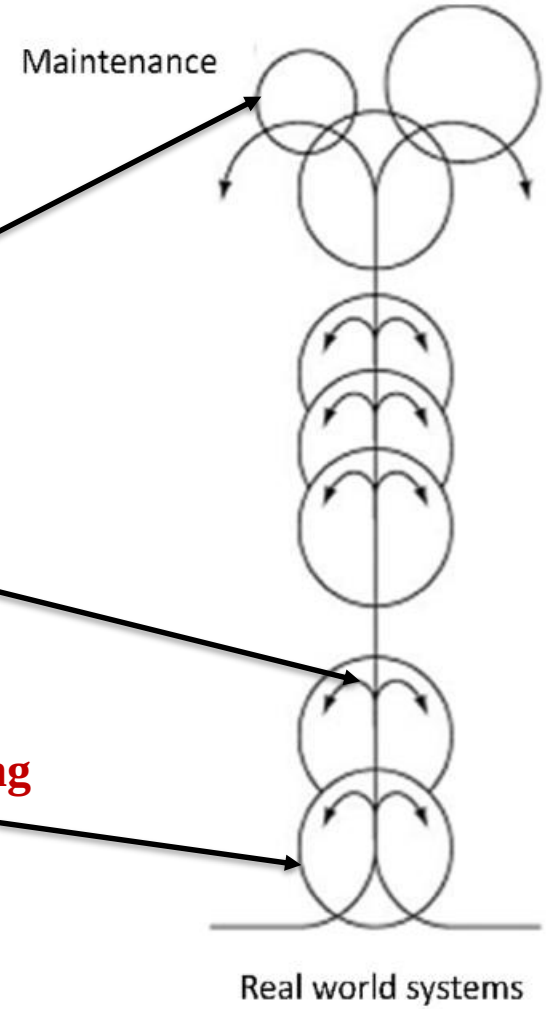
- Comparison between phases of traditional and object-oriented approaches: (Conti…)

	Traditional approaches	Object-oriented approaches
Implementation and Testing phase	Implementing processes, Functions, Unit, Integration and System testing.	Implementing objects and interactions amongst the objects. Unit, Integration and System testing.
Documentation	Many documents are produced at the end of each phase.	A document may or may not be produced at the end of each phase.

CE: 1.6.1 Fountain Model

- It provides a clear representation of iterations and overlapping phases.
- This model emphasizes on the reusability of the source code.

CE: 1.6.1 Fountain Model (Conti…)



The Maintenance phase is the smallest in the Fountain Model.

It is the iteration within the phases.

It is the overlapping phases.

CE: 1.6.1 Fountain Model (Conti…)

- In this model, there is no need to freeze the requirement in the early phases of the software development.
- The system is developed in terms of classes and the phases of the software development life cycle.
- **But, this model may follow an undisciplined approach in which the developer moves randomly between phases.**

CE: 1.6.2 Rational Unified Process (RUP) Model

- It is a software development process from Rational, a division of IBM.

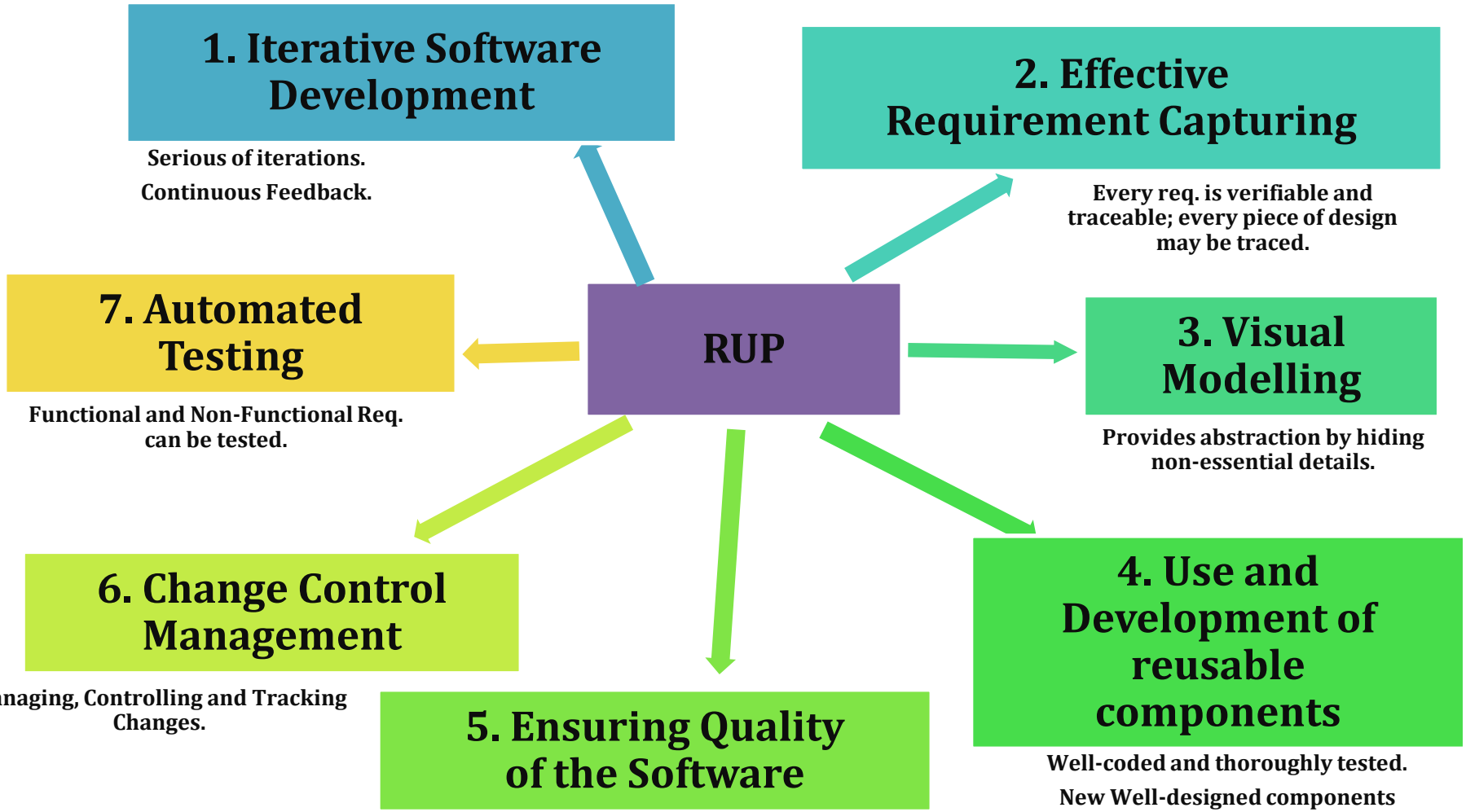
- The RUP forms provides basic guidelines to use Unified Modelling Language (UML).

What is UML?

✓ The UML is a popular standard for visually modelling elements of the system and is governed by Object Management Group (OMG).

✓ The UML is a language for *creating*, *visualizing* and *documenting* various diagrams that represents different aspects of the software.

CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)



Key features of RUP.

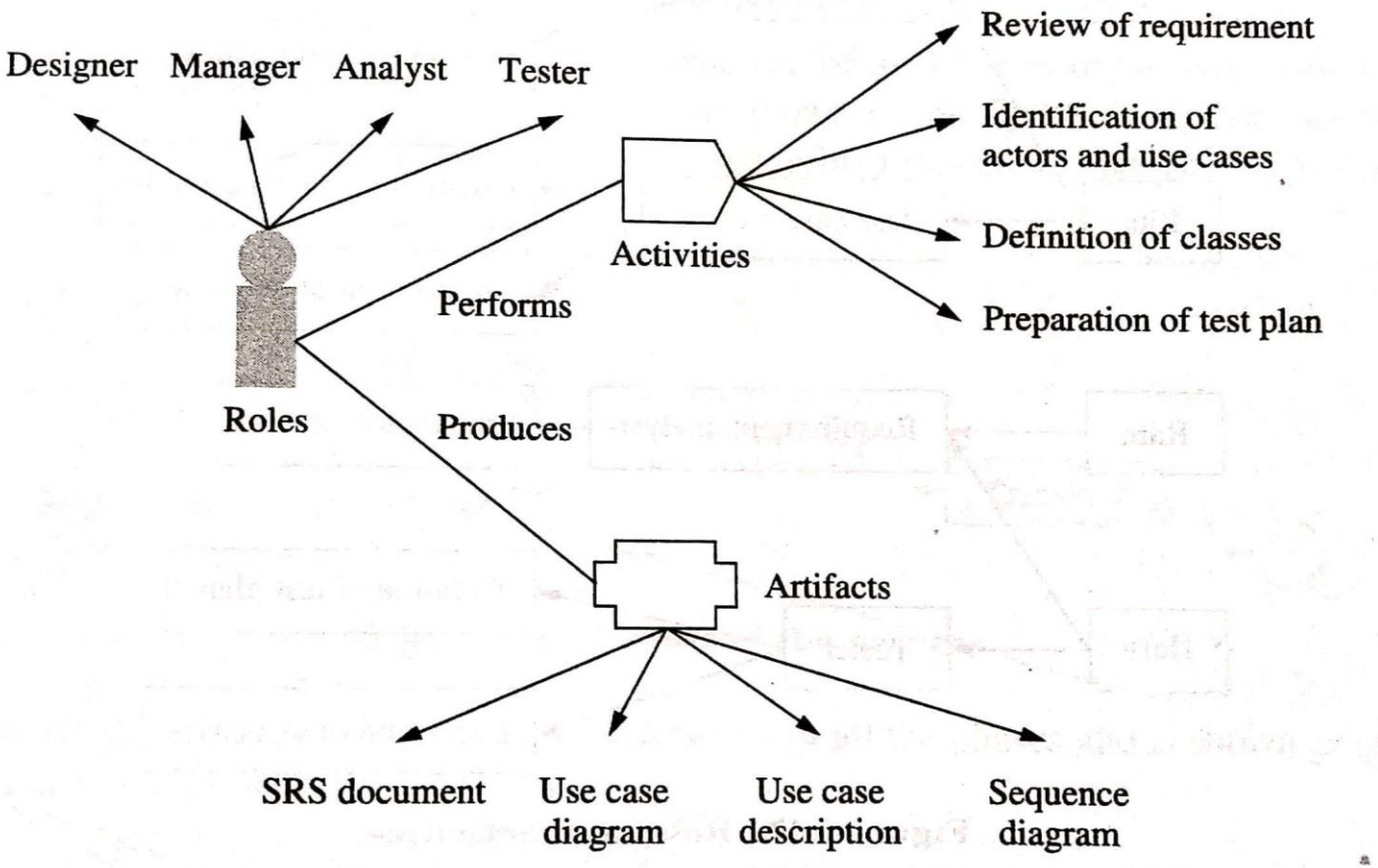
CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)

- The process can be divided into structures:
  1. Static Structure
  2. Dynamic Structure

1. Static Structure:

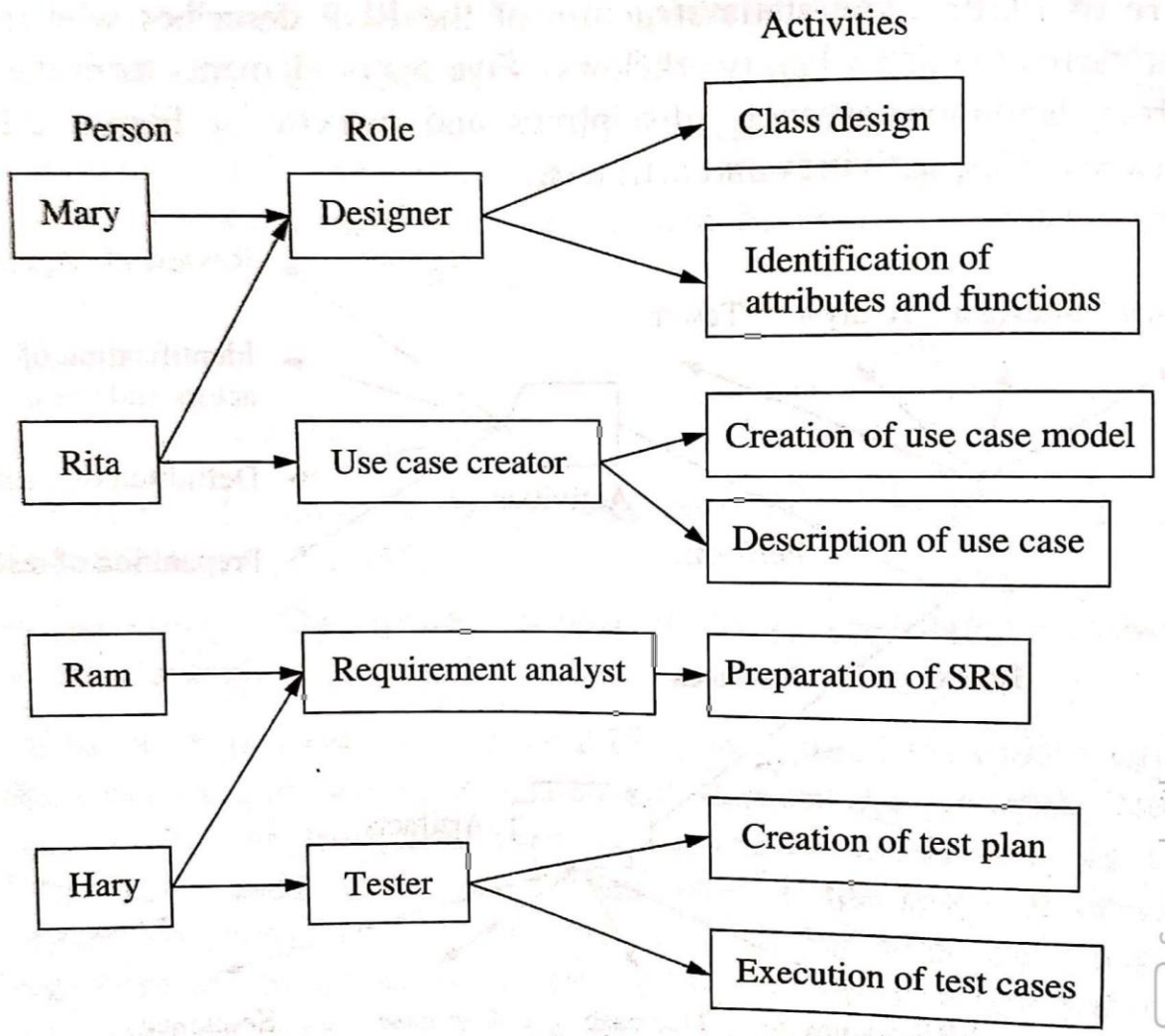
- It describes who, how, what and when.
- It provides five major elements form that the process descripts are :
  1. roles, → **who**
  2. activities, → **how**
  3. artifacts (work of art), → **what**
  4. disciplines and
  5. workflows. → **when**

CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)



Relationship between roles, activities and artifacts

CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)



Roles and activities

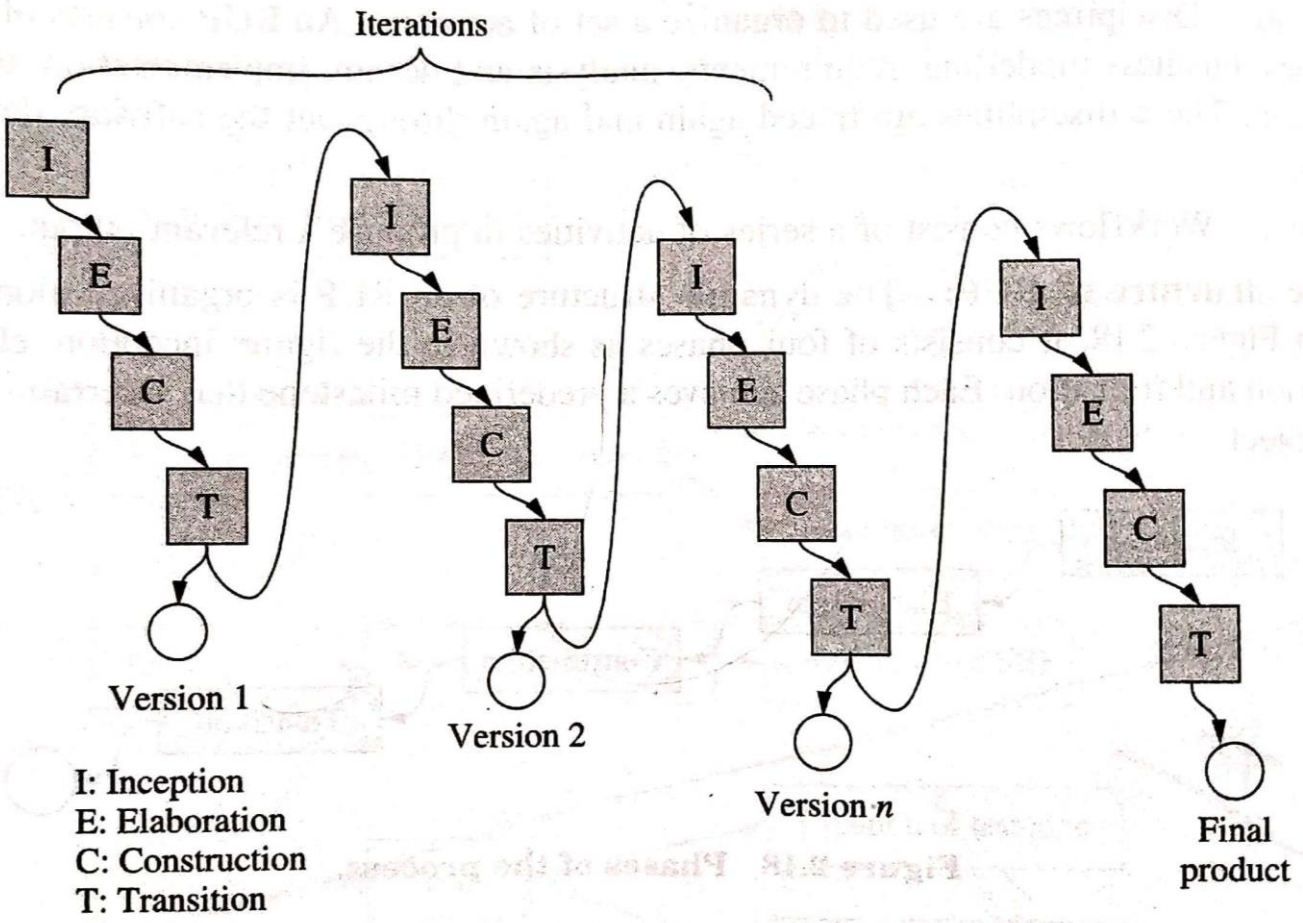
CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)

2. Dynamic Structure:

- The process that can be viewed in terms of iterative development.

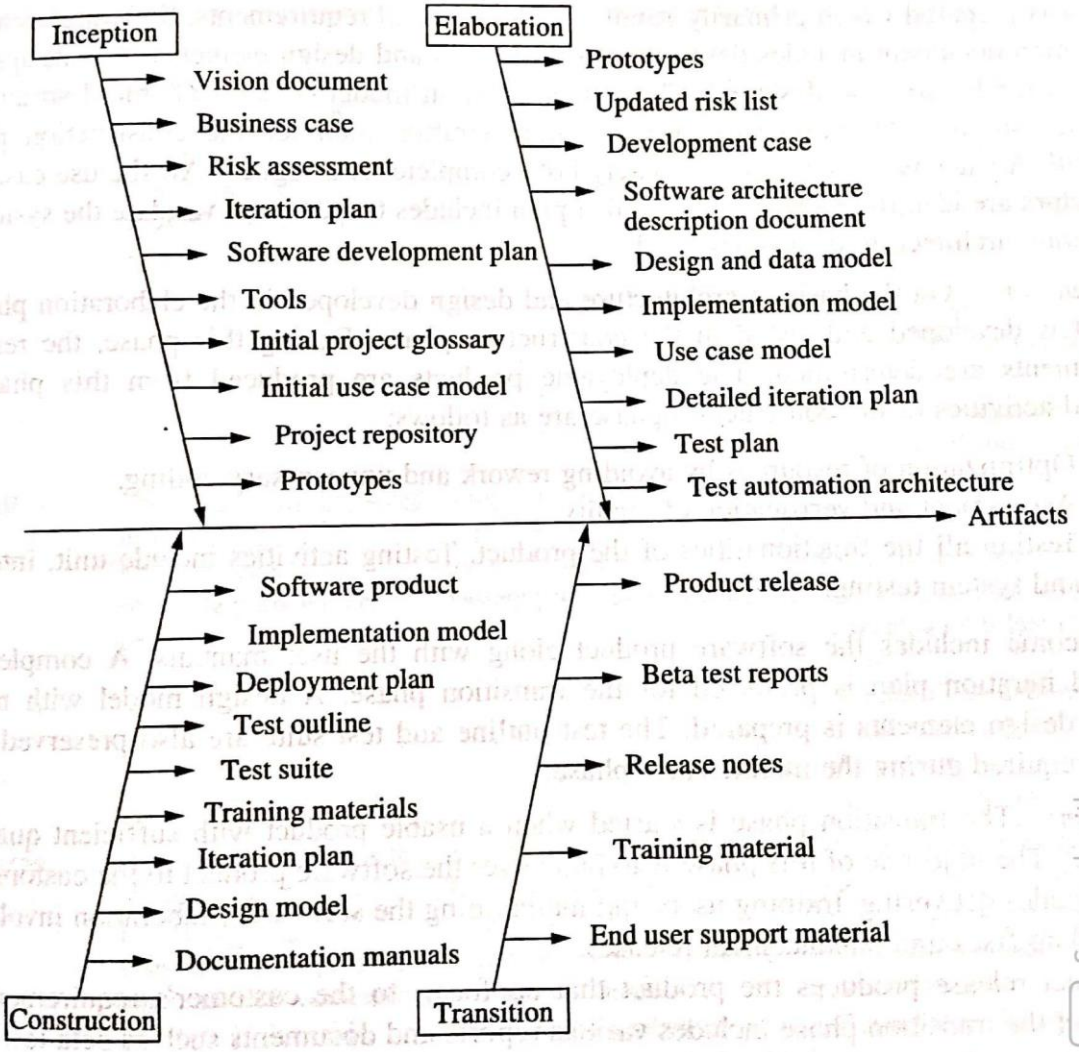
- It has four phases:
  1. Inception
  2. Elaboration
  3. Construction
  4. Transition

CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)



Iterative RUP

CE: 1.6.2 Rational Unified Process (RUP) Model (Conti…)



Fish bone diagram showing the artifacts produced.


Thank You