Life Cycle Models

Edit with WPS Office

Life Cycle Models

- Many life cycle models have been **proposed**.
- We will confine our attention to a few important and **commonly used** models.
  - classical waterfall model
  - iterative waterfall model,
  - Evolutionary model,
  - Prototyping model,
 - Agile model (rapid), and
  - spiral model.

Edit with WPS Office

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.

Edit with WPS Office

Classical Waterfall Model

Feasibility Study → Req. Analysis → Design → Coding → Testing → Maintenance
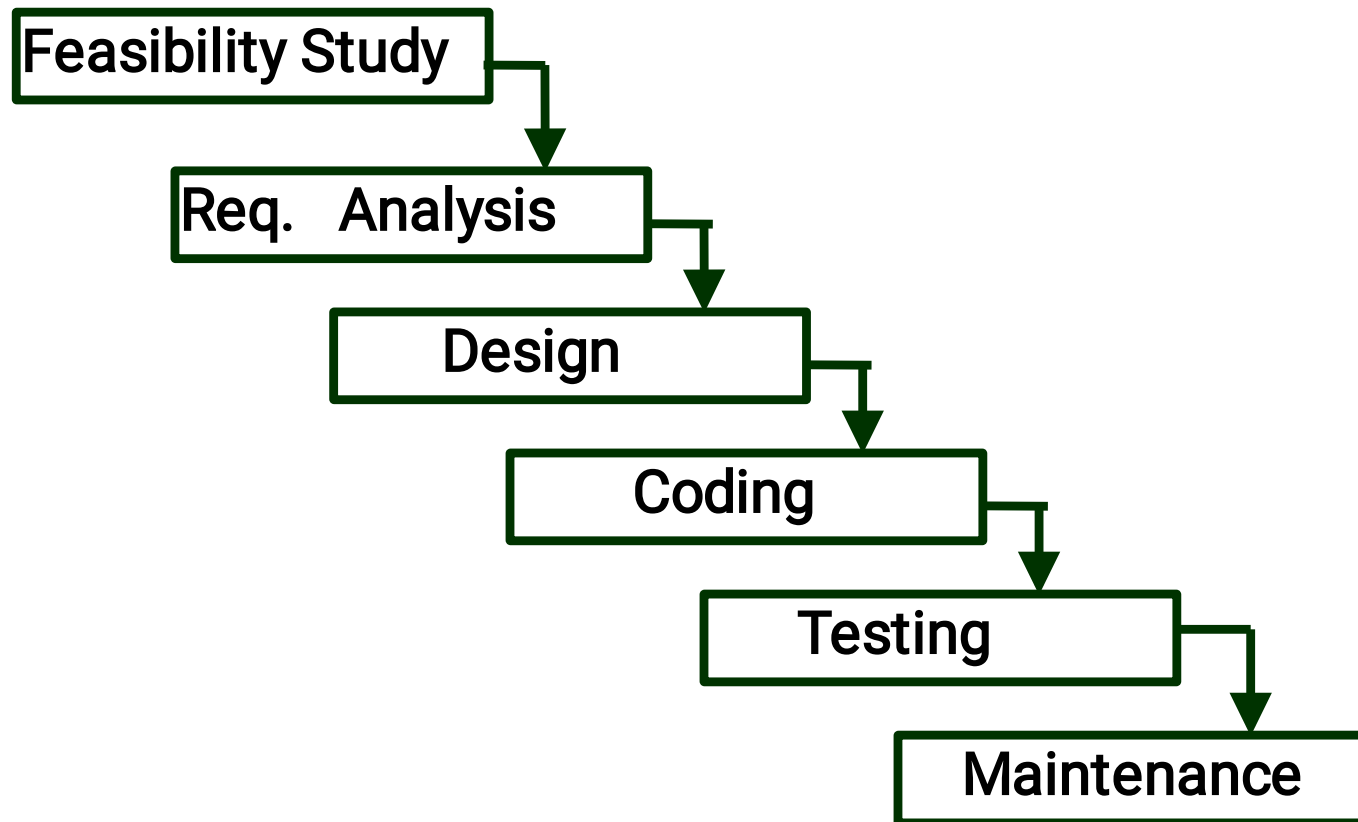
Relative Effort for Phases

- **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.

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.

Edit with WPS Office

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
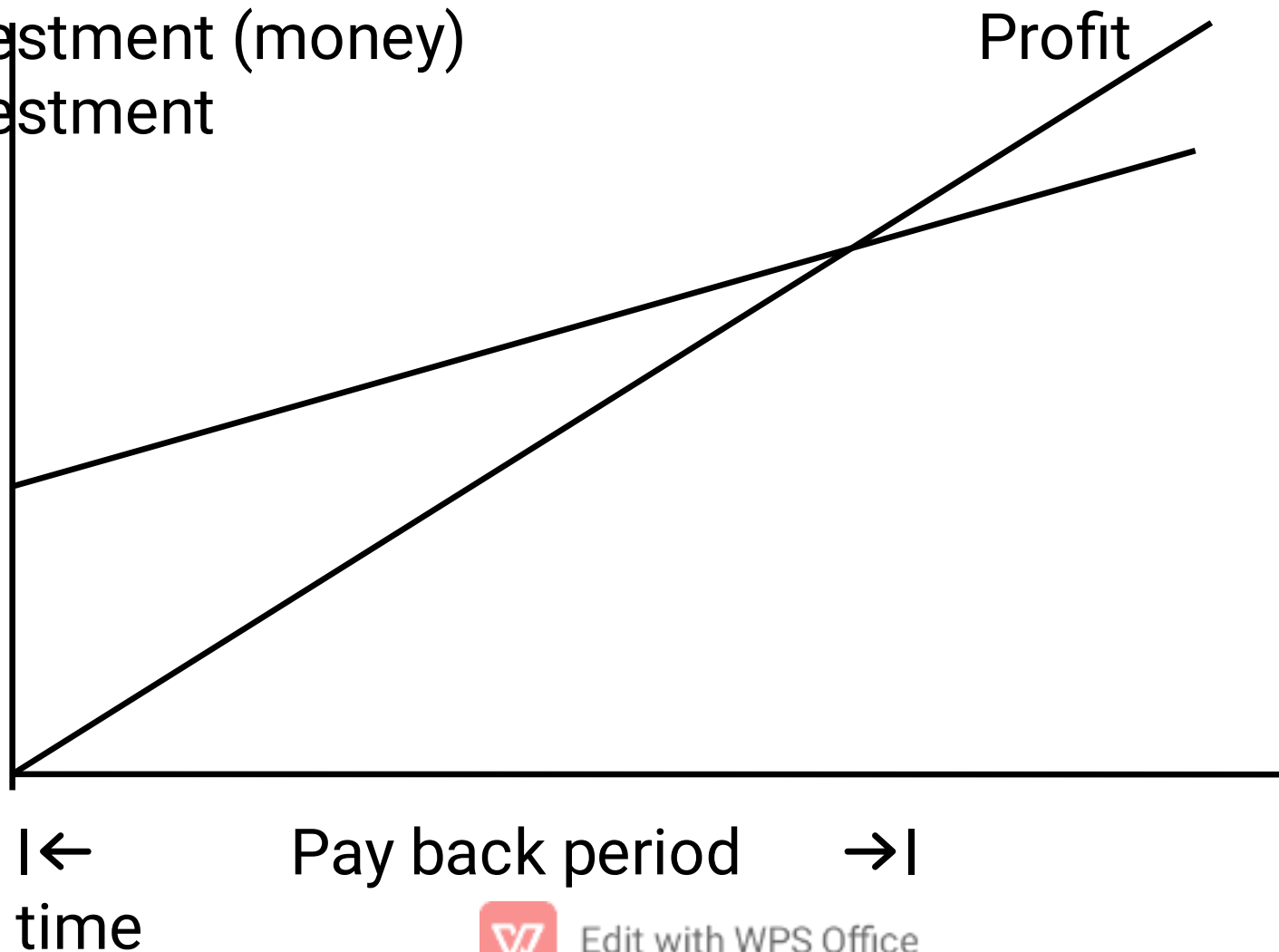   * development time.

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.

Break even analysis

- Investment (money)
Investment

Profit

I←   Pay back period   →I
time

Requirements Analysis and Specification (Elicitation and Gathering)

- ## 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.

Edit with WPS Office

Goals of Requirements Analysis

- **Collect all related data from the customer:**
  - analyze the collected data to clearly understand what exactly 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/discuss 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.

Edit with WPS Office

Requirements Analysis (CONT.)

- **Ambiguities and contradictions:**
  - must be identified,
  - resolved by discussions with the customers (at every level).
- **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, <u>software architecture</u> is derived from the SRS document.

- Two design approaches:
  - Traditional/ Structured approach,
  - object oriented approach.

Traditional/ Structured Approach

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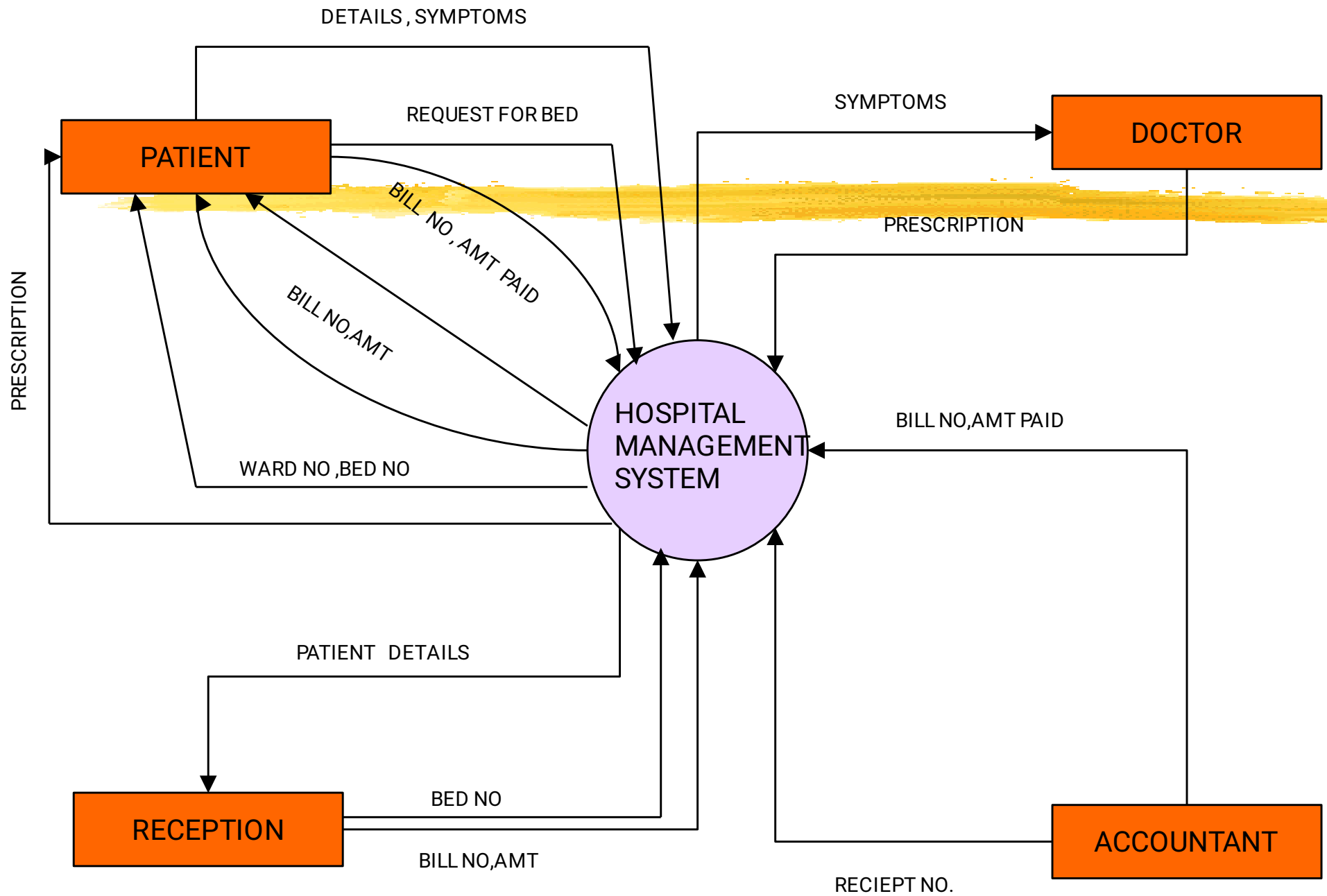
Structured Analysis Activity

- **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 .... and so on.
  - ....

Edit with WPS Office

Structured Analysis

- 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).

Edit with WPS Office

DETAILS , SYMPTOMS

REQUEST FOR BED

SYMPTOMS

DOCTOR

PATIENT

BILL NO ,AMT PAID

PRESCRIPTION

PRESCRIPTION

BILL NO,AMT

HOSPITAL MANAGEMENT SYSTEM

BILL NO,AMT PAID

WARD NO ,BED NO

PATIENT DETAILS

BED NO

RECEPTION

ACCOUNTANT

BILL NO,AMT

RECIEPT NO.

DFD LEVEL 0 (context diagram)

22

Structured Design

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

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

Edit with WPS Office

LEVEL 1

HOSPITAL MANAGEMENT SYSTEM
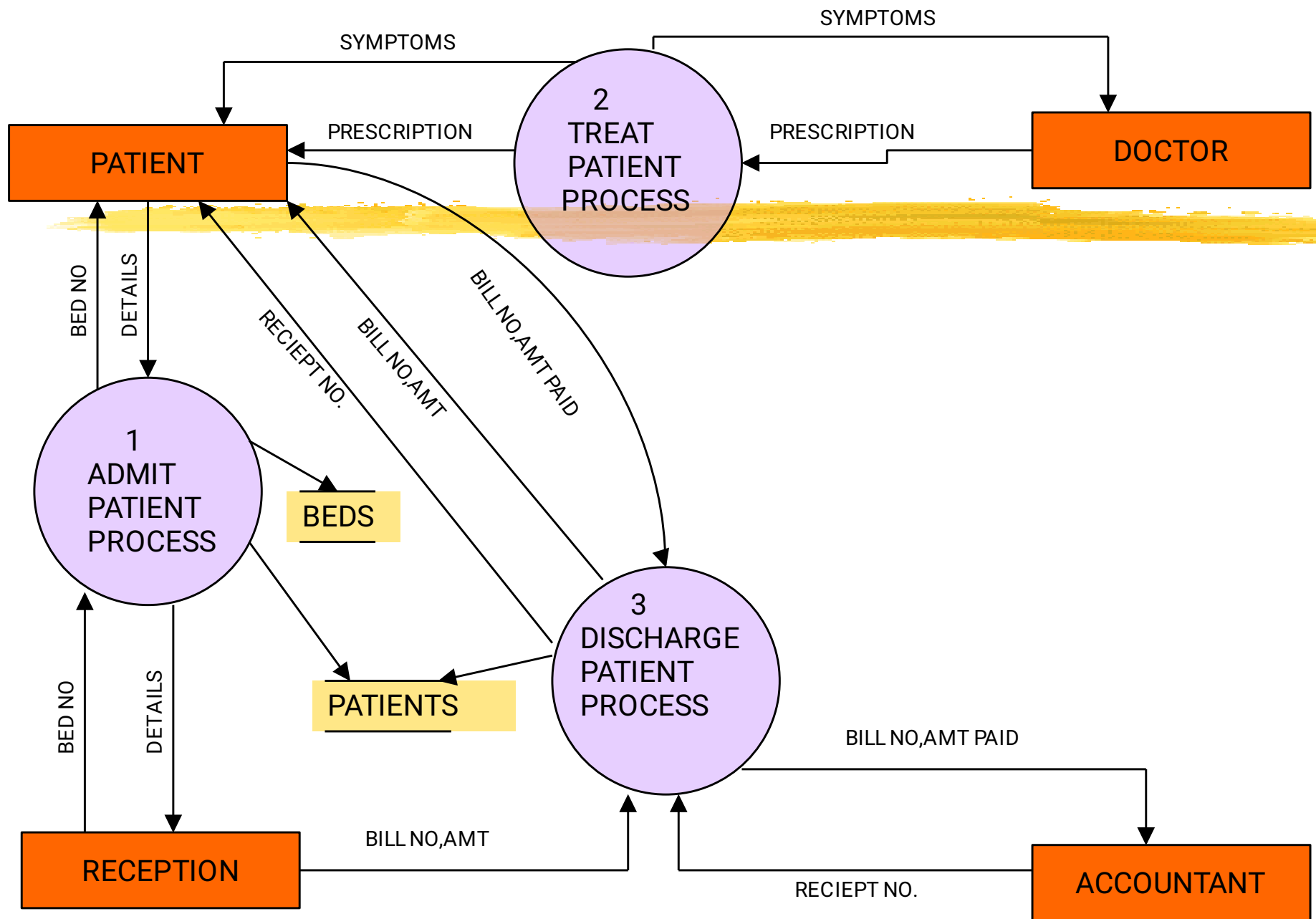
Object Oriented Design

- **First identify various objects (real world entities) occurring in the problem:**
  - identify the relationships among these objects. Also identify their behaviour (functionality) and their communication.
  - For example, the objects in a pay-roll system may be:
    - * employees, managers, pay-roll register, Departments, etc.

Edit with WPS Office

What is UML ?

- **Unified Modeling Language**

  Used to visualizing, specifying and constructing the software systems.

- **Unified because it …**

  Combines main preceding OO methods (**OOAD by Grady Booch, OMT by Jim Rumbaugh and OOSE by Ivar Jacobson**).
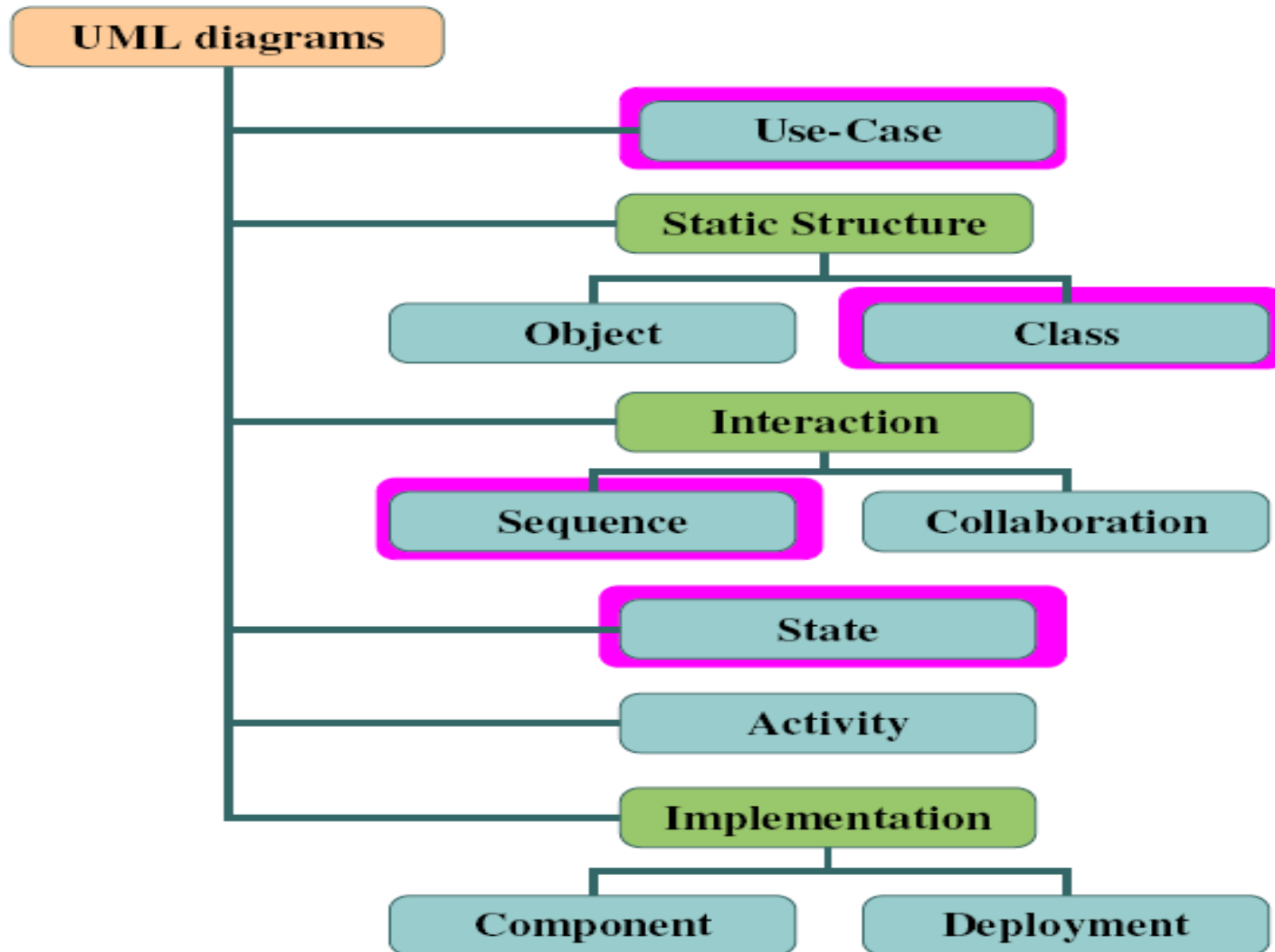
- **Modeling because it is …**

  Primarily used for visually **modeling** the systems. Many system views are supported by appropriate models.
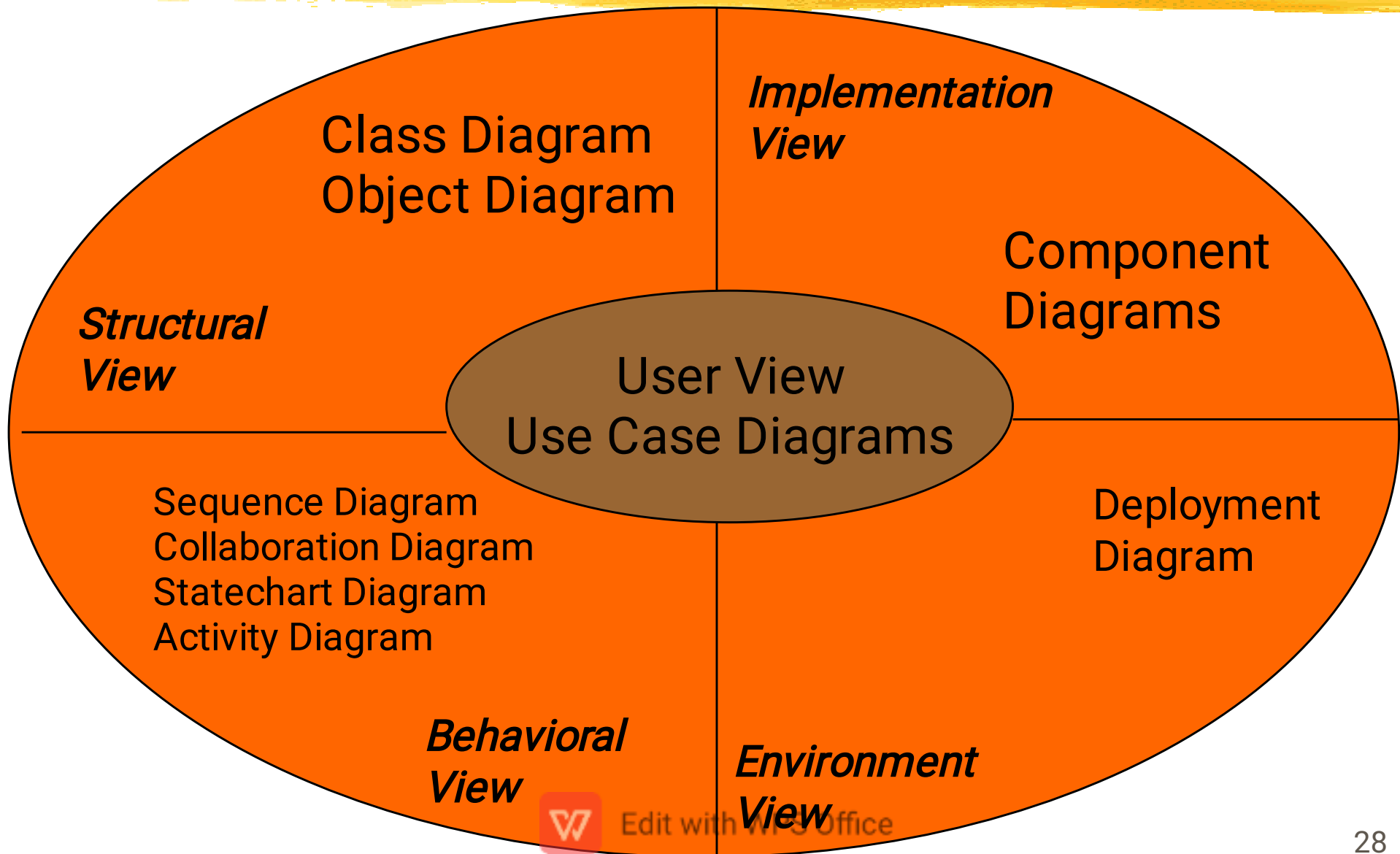
- **Language because …**

  It offers a **syntax** through which the system can be modeled.

Edit with WPS Office

UML diagrams

```
UML diagrams
    ├── Use-Case
    ├── Static Structure
    │       ├── Object
    │       └── Class
    ├── Interaction
    │       ├── Sequence
    │       └── Collaboration
    ├── State
    ├── Activity
    └── Implementation
            ├── Component
            └── Deployment
```

Edit with WPS Office

Model and Views of UML

Structural View

Class Diagram
Object Diagram

Implementation View

Component Diagrams

User View
Use Case Diagrams

Sequence Diagram
Collaboration Diagram
Statechart Diagram
Activity Diagram

Deployment Diagram

Behavioral View

Environment View

UML Diagrams

- **Use case diagrams:**
  - represent the **functions** of a system from the **user's (actor's) point of view**.
- **Sequence diagrams**:
  - are a temporal representation of **objects and their interactions**.
- **Collaboration diagrams**:
  - spatial representation of **objects, links**, and **interactions**.
- **Object diagrams :**
  - represent **objects** and their **relationships**.
- **Class diagrams:**
  - represent the static structure in terms of classes and relationships.
- **State-chart diagrams**:
  - represent the **behavior of a system** in terms of states.
- **Activity diagrams:**
  - represent the **behavior of an activity (flow of an operation)** as a set of actions or tasks
- **Component diagrams:**
  - represent various **components** of an application
- **Deployment diagrams:**
  - represent the deployment of physical **components** on particular pieces of **hardware**

Edit with WPS Office

Use Case Diagrams..

Use case diagram (example) for Library system
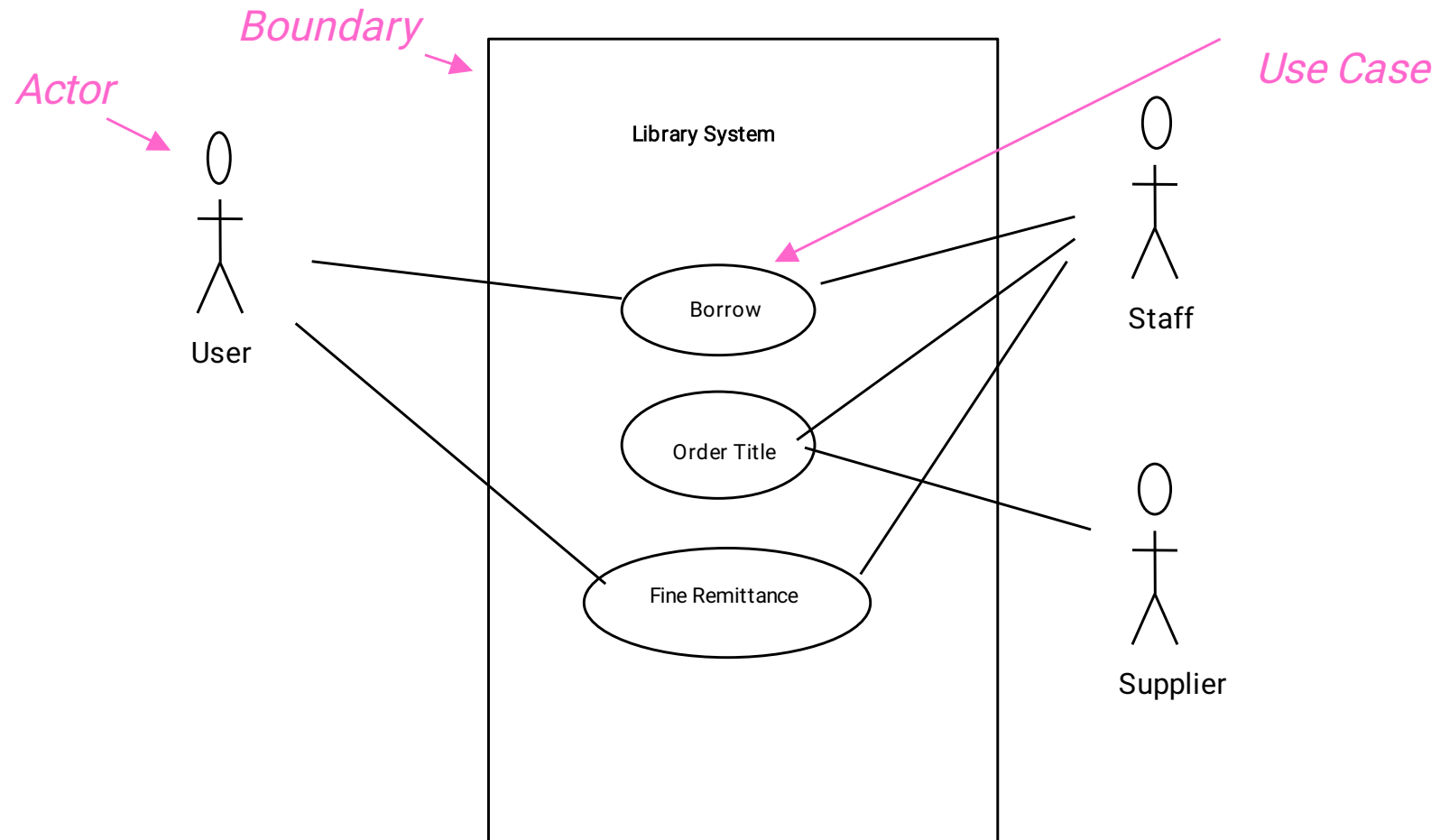
A generalized description of how a system is in use.

• Provides an overview of the intended functionality of the system
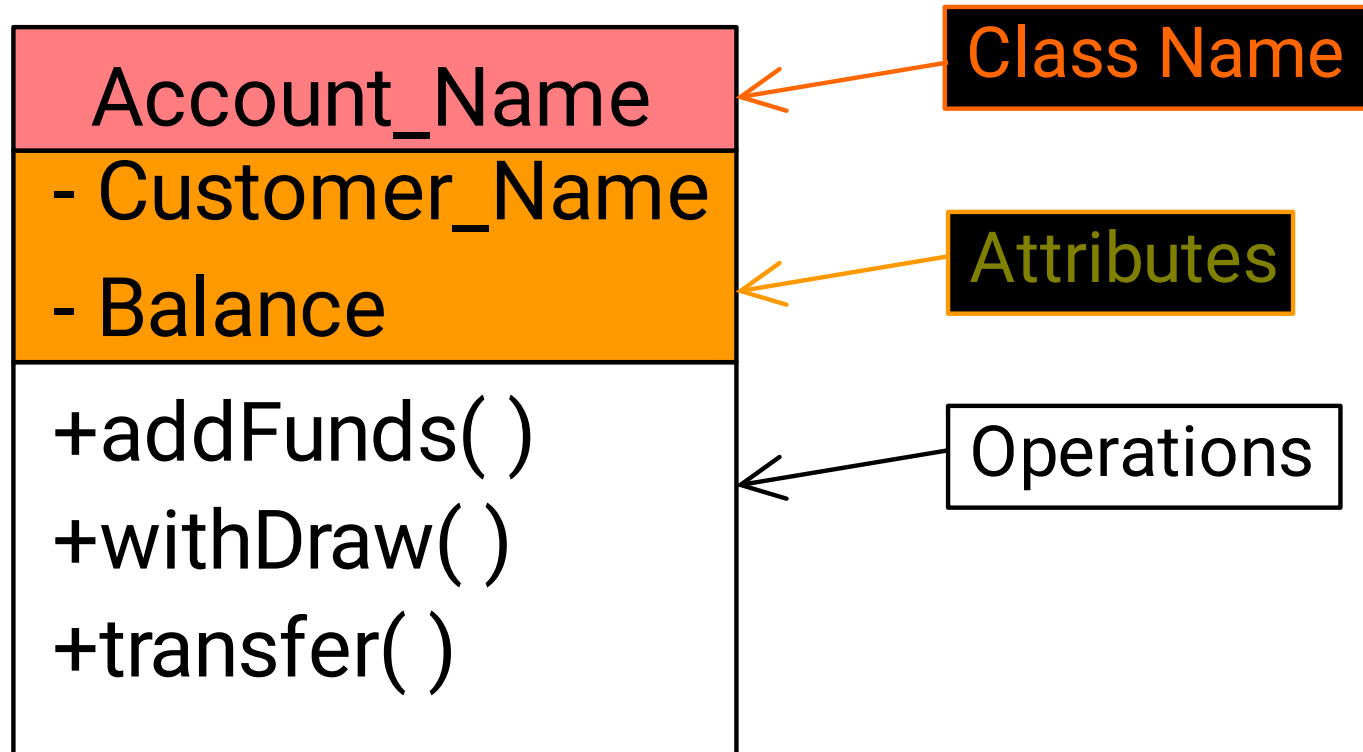
Use Cases

Definition:
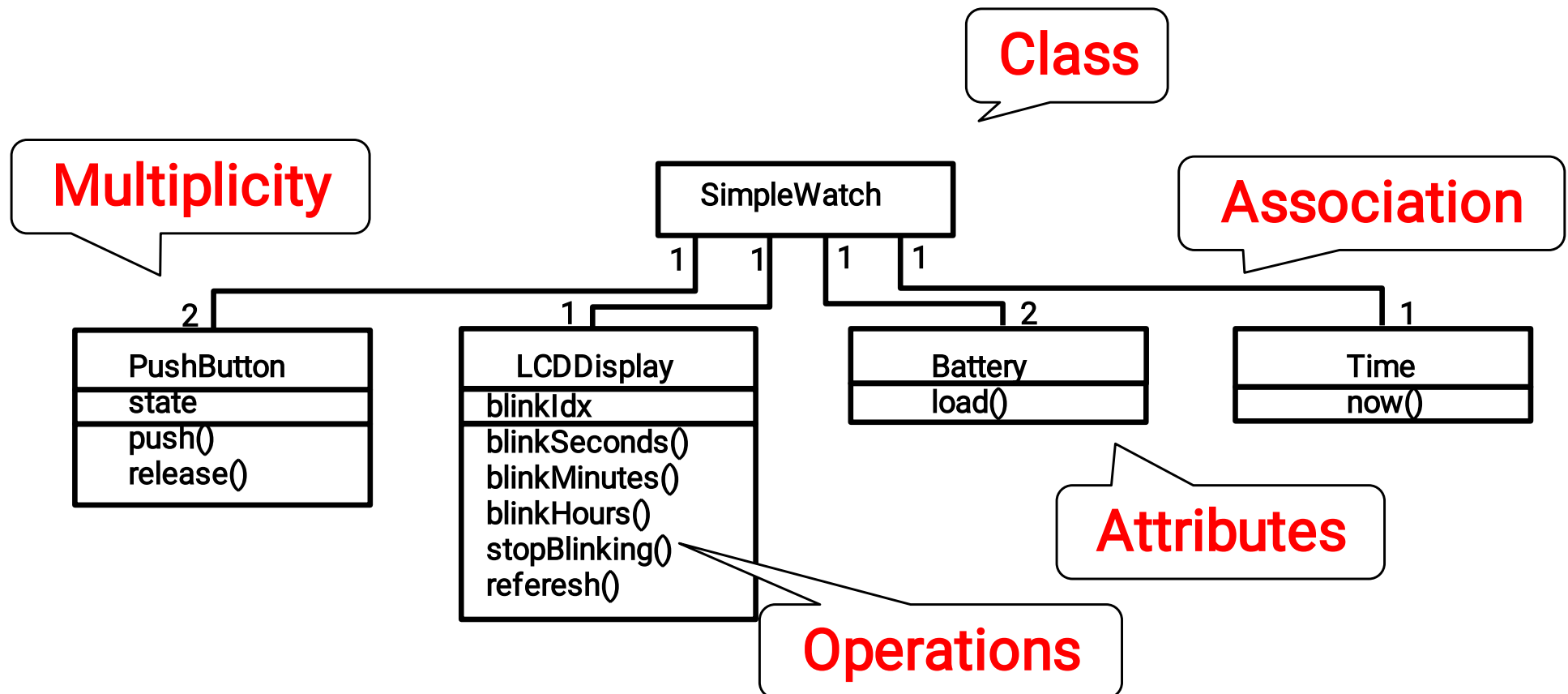
Use Case = Closed loop interaction with the user

The refinement process of the top down approach is replaced by listing all use cases, or: "write down everything the system is supposed to do"

Edit with WPS Office

An example of Class

Account_Name
- Customer_Name - Balance
+addFunds( ) +withDraw( ) +transfer( )

Class Name

Attributes

Operations

32

UML First Pass: Class Diagrams

Class

Multiplicity

Association

Attributes

Operations

```
                              SimpleWatch
                     1    1    1    1

  2                       1                       2                 1
PushButton              LCDDisplay              Battery           Time
state                   blinkIdx                load()            now()
push()                  blinkSeconds()
release()               blinkMinutes()
                        blinkHours()
                        stopBlinking()
                        referesh()
```

Class diagrams represent the structure of the system

Sequence Diagram
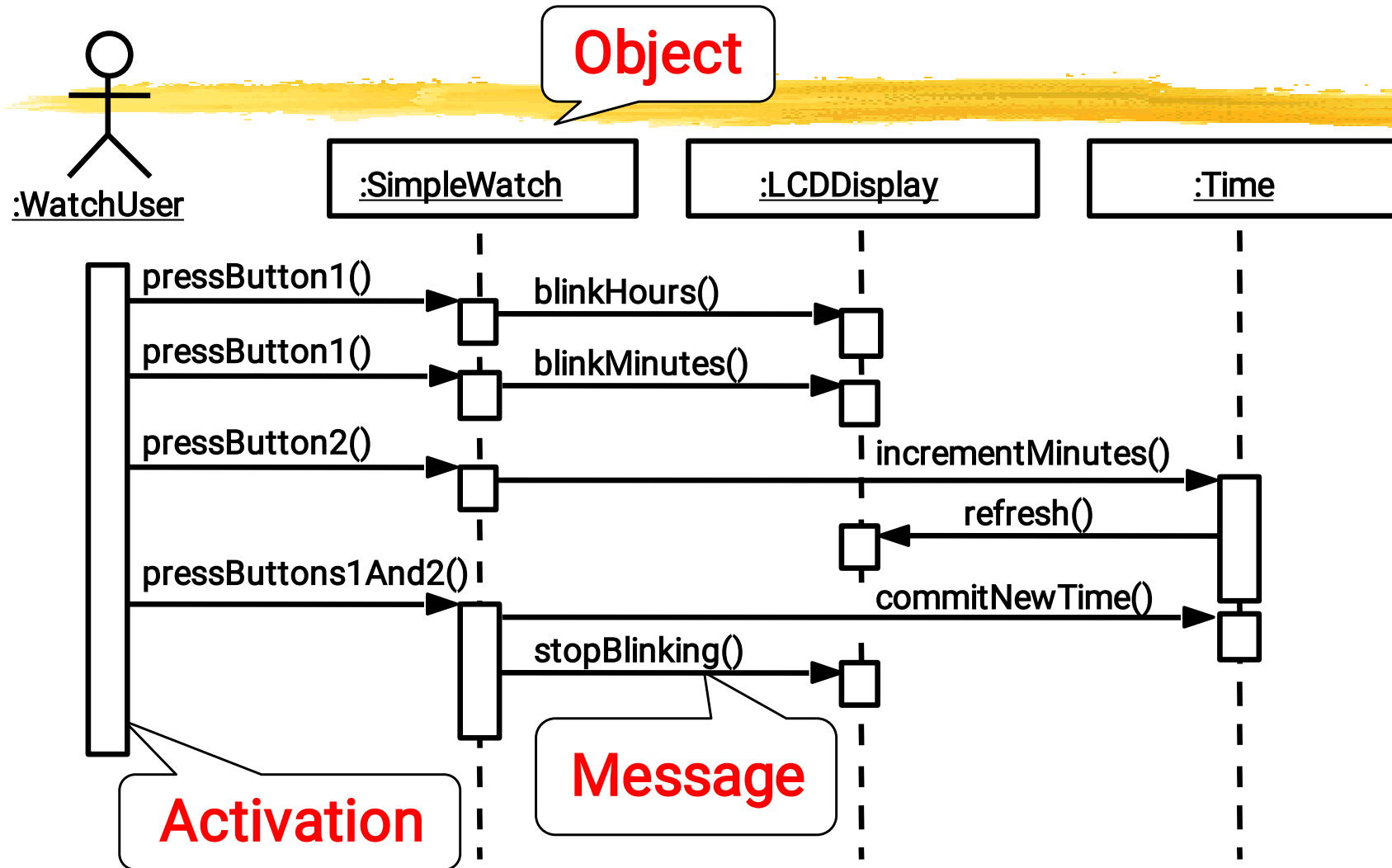
Object Oriented Design (CONT.)

- **Object structure**
  - further refined to obtain the detailed design.

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

SDLC Activities (in OO Paradigm)

Independent of how they are organized, the following activities are involved in the development of software:

1.	Requirements analysis	The requirements for the software product are determined and documented.
2.	Architectural design	The architecture of the solution is determined. This breaks the solution into different components that are allocated to one or more processing resources.
3.	Component design	For each component, classes are identified. Responsibilities are assigned to classes.
4.	Detailed design	Methods and attributes are defined for classes. Detailed algorithms for the methods are defined.
5.	Coding	The individual methods are coded in the target programming language.
6.	Unit Test	The methods are tested in isolation, and as a class.
7.	Integration test	The methods and classes are tested together to verify that they work together and meet the requirements.
8.	Acceptance test	The product as a whole is tested against its requirements to demonstrate that the product meets its requirements.
9.	Installation	The product is installed in its end use (production) environment.
10.	Maintenance	corrections are made to the product.

Implementation

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

Implementation

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

Edit with WPS Office

Implementation (CONT.)

- **The purpose of unit testing:**
  - test whether 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 also tested.

Edit with WPS Office

Integration and System Testing

System Testing

- **After all the modules have been successfully integrated and tested:**
  - **system testing is carried out.**

- **Goal of system testing:**
  - ensure that the developed system **functions** according to its requirements as specified in the SRS document.

Performance testing and acceptance testing

- Load testing

- Reliability testing

- Security testing…   etc.

- Alpha testing

- Beta testing

Edit with WPS Office

43

Maintenance

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

Maintenance (CONT.)

- **Corrective maintenance:**
  - *Correct errors* which were not discovered during the product development phases.
- **Preventive maintenance:**

  take *preventive measures* so that error/fault cannot be introduced.
- **Perfective maintenance:**
  - Improve implementation of the system and
  - *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.

Edit with WPS Office

- The classical waterfall model can be considered to be a *theoretical way of developing software.*

- But all other life cycle models are essentially derived from the classical waterfall model.

When to use the Waterfall Model

- Requirements are very well known,
- Product definition is stable,
- Technology is understood,
- New version of an existing product,
- Porting an existing product to a new platform.

Waterfall Strengths

- Easy to understand, easy to use,
- Provides structure to inexperienced staff,
- Milestones are well understood,
- Sets requirements stability,
- Good for management control (plan, staff, track)

Edit with WPS Office

Waterfall Deficiencies

- All requirements must be **known clearly** beforehand,
- Deliverables created for each phase are considered frozen – **reduces flexibility**
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases

Little opportunity for customer to preview the system (until it may be too late)

Iterative Waterfall Model

- **Classical waterfall model is idealistic:**

  - assumes that no defect is introduced during any development activity.

  - But in practice:

    * defects do get introduced in almost every phase of the life cycle.
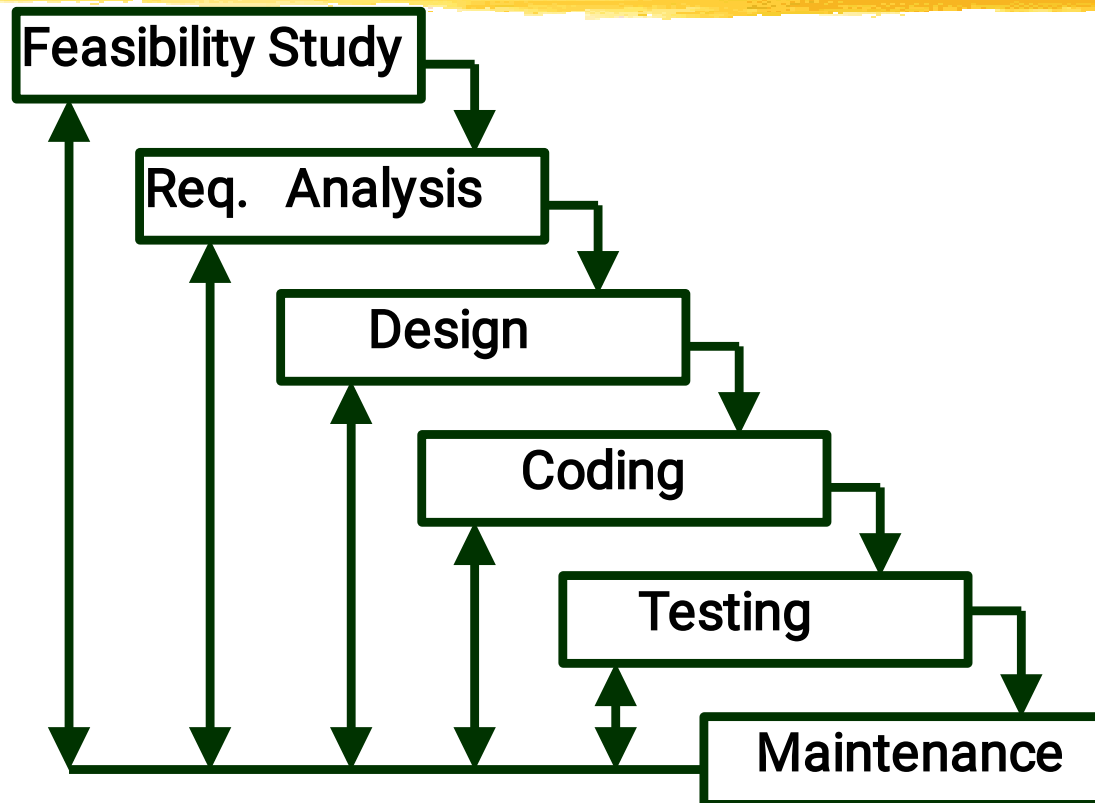
Iterative Waterfall Model (CONT.)

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

Edit with WPS Office

Iterative Waterfall Model (CONT.)

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

- **Therefore we need *feedback paths* in the classical waterfall model.**

Iterative Waterfall Model (CONT.)

Iterative Waterfall Model (CONT.)

- **Errors should be detected**
  - in the same phase in which they are introduced.
- **For example:**
  - **if a design problem is detected in the design phase itself,**
    - the problem can be taken care of, much more easily
    - than if it is identified at the end of the integration and system testing phase.

Phase containment of errors

- <u>Reason:</u> 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.

Edit with WPS Office

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

- Developers build a prototype during the requirements phase,

- Prototype is evaluated by end users,

- Users give corrective feedback,

- Developers further refine the prototype,

- When the user is satisfied, the prototype code is brought up to the standards needed for a final product.

Edit with WPS Office

Reasons for developing a prototype

- Illustrate to the customer:
  - input data formats, interactions and reports.
- Examine technical issues associated with product development:
  - Often major design decisions depend on issues like:
    - * response time of a sub system (e.g. hardware controller),
    - * efficiency of a function (e.g. sorting

Prototyping Model (CONT.)

- **The next reason for developing a prototype is:**
  - it is impossible to ``get it right" at the first time itself,
  - So we must plan and get ready to throw away, the first product
    - * if we want to develop a good quality product.

Edit with WPS Office

Prototyping Model (CONT.)

- **Start with approximate requirements.**

- **Carry out a quick design.**

- **Prototype model is built using several short-cuts:**

  - Short-cuts might involve using inefficient, inaccurate, or dummy functions.

Edit with WPS Office

Prototyping Model (CONT.)

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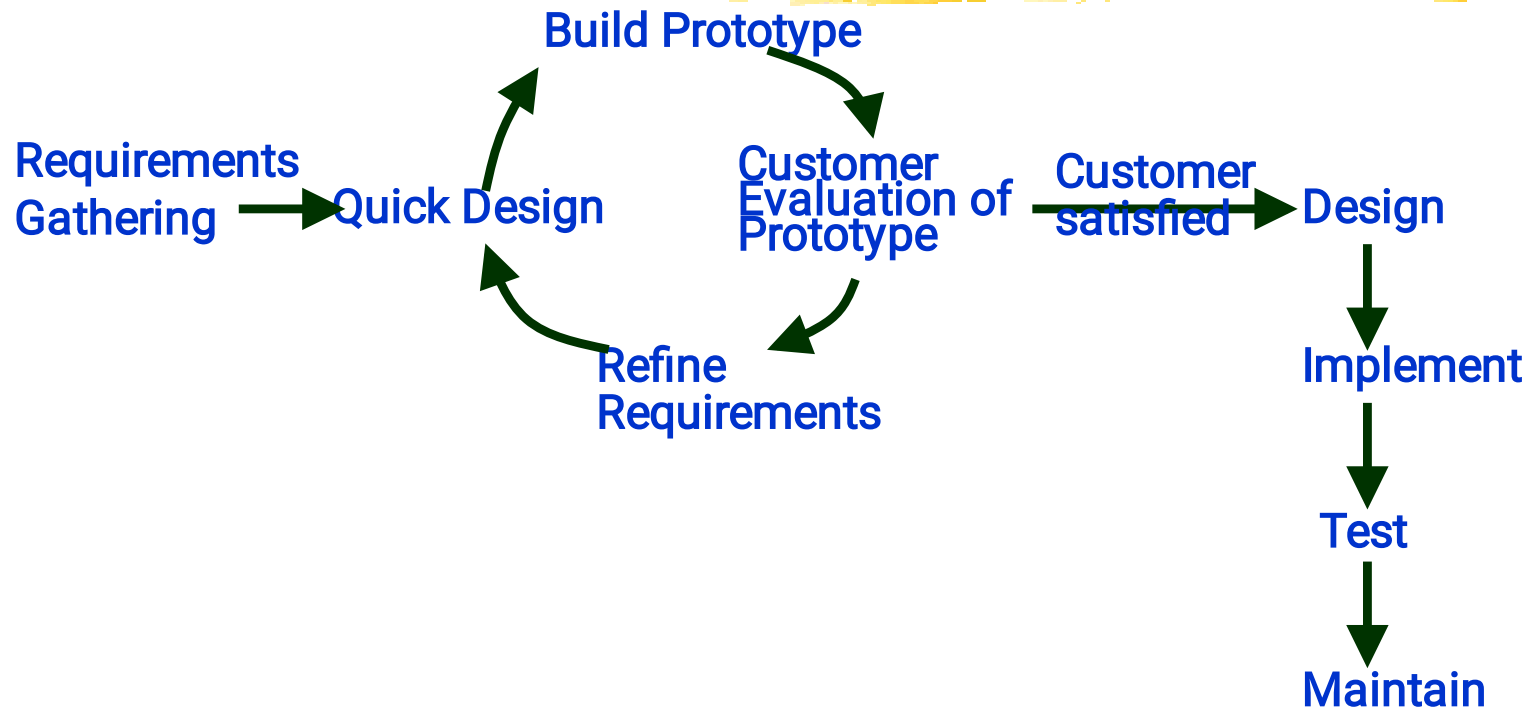- **The actual system is developed using the classical waterfall approach.**

Edit with WPS Office

Prototyping Strengths

- Customers can "see" the system requirements as they are being gathered,
- Developers learn from customers,
- A more accurate end product can be built,
- Unexpected requirements accommodated,
- Allows for flexible design and development,
- Interaction with the prototype stimulates awareness of additional needed functionality.

Edit with WPS Office

62

Prototyping Model (CONT.)

Build Prototype

Requirements Gathering → Quick Design → Customer Evaluation of Prototype → Customer satisfied → Design

Refine Requirements

Design → Implement → Test → Maintain

63

Prototyping Model (CONT.)

- **Requirements analysis and specification phase becomes redundant:**
  - final working prototype (with all user feedbacks incorporated) serves as an animated (active) 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.

Edit with WPS Office

Evolutionary Model

- Evolutionary model (i.e. incremental model or successive versions model):
  - The system is broken down into several sub systems/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:
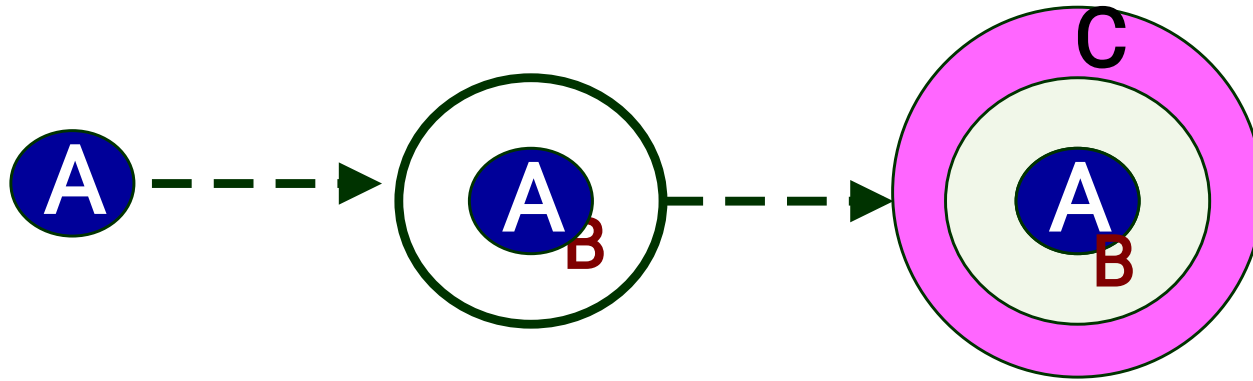 - by adding new functionalities in successive versions.

Edit with WPS Office

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.

Edit with WPS Office

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.

Edit with WPS Office

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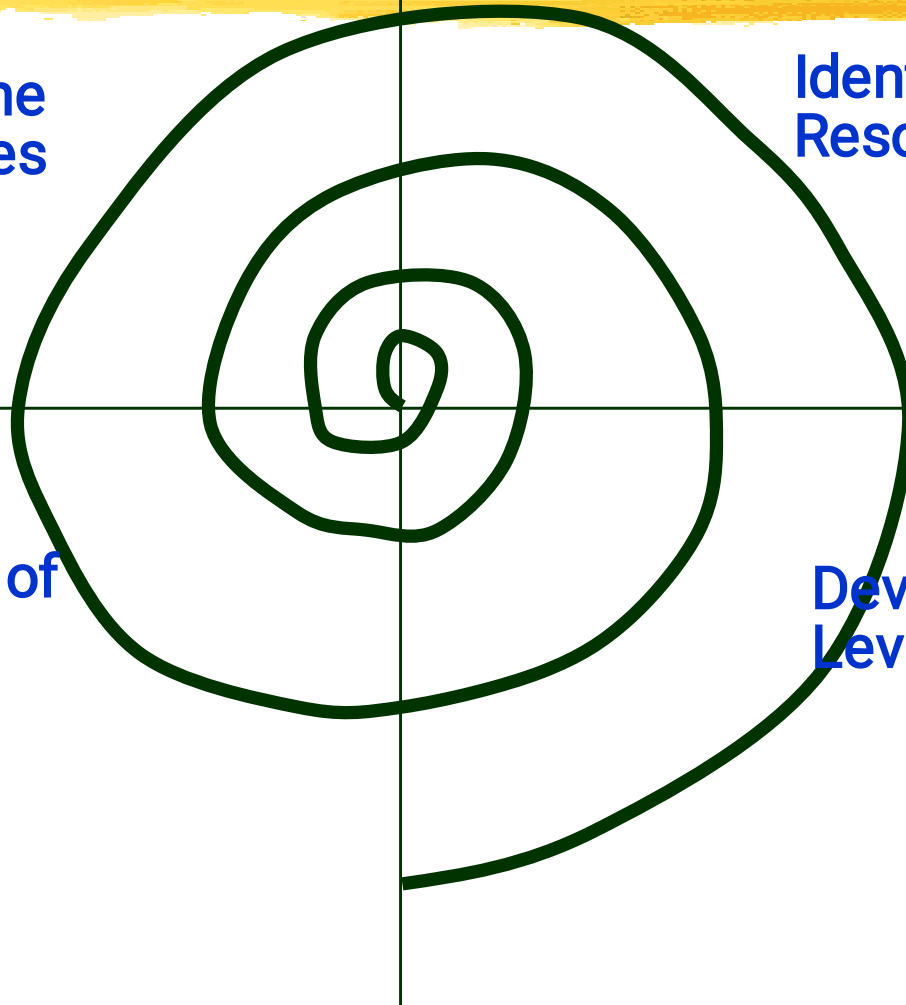- **Each loop in the spiral is split into four sectors (quadrants).**

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.

Edit with WPS Office

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.

Edit with WPS Office

Spiral Model as a meta model

- **Includes 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.

Other process models…

- RUP process model,
- Agile process model,
- Xtreem programming process model etc…

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

- **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**.

Edit with WPS Office

Summary

- Software engineering is:
  - systematic collection of decades of programming experience and software development process,
  - together with the innovations made by researchers.

Summary

- A fundamental necessity while developing any large software product:
  - adoption of a life cycle model.

Summary

- Adherence to a software life cycle model:
  - helps to do various development activities in a systematic and disciplined manner.
  - also makes it easier to manage a software development effort.