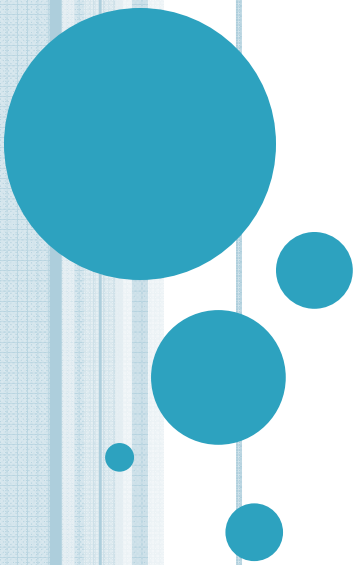
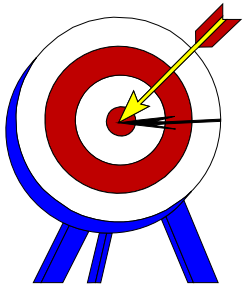
OBJECT-ORIENTED SYSTEMS DEVELOPMENT: USING THE UNIFIED MODELING LANGUAGE



Object-Oriented Systems Development Life Cycle



GOALS

- The software development process
- Building high-quality software
- Object-oriented systems development
- Use-case driven systems development
- Prototyping
- Rapid application development
- Component-based development
- Continuous testing and reusability

THE SOFTWARE DEVELOPMENT PROCESS

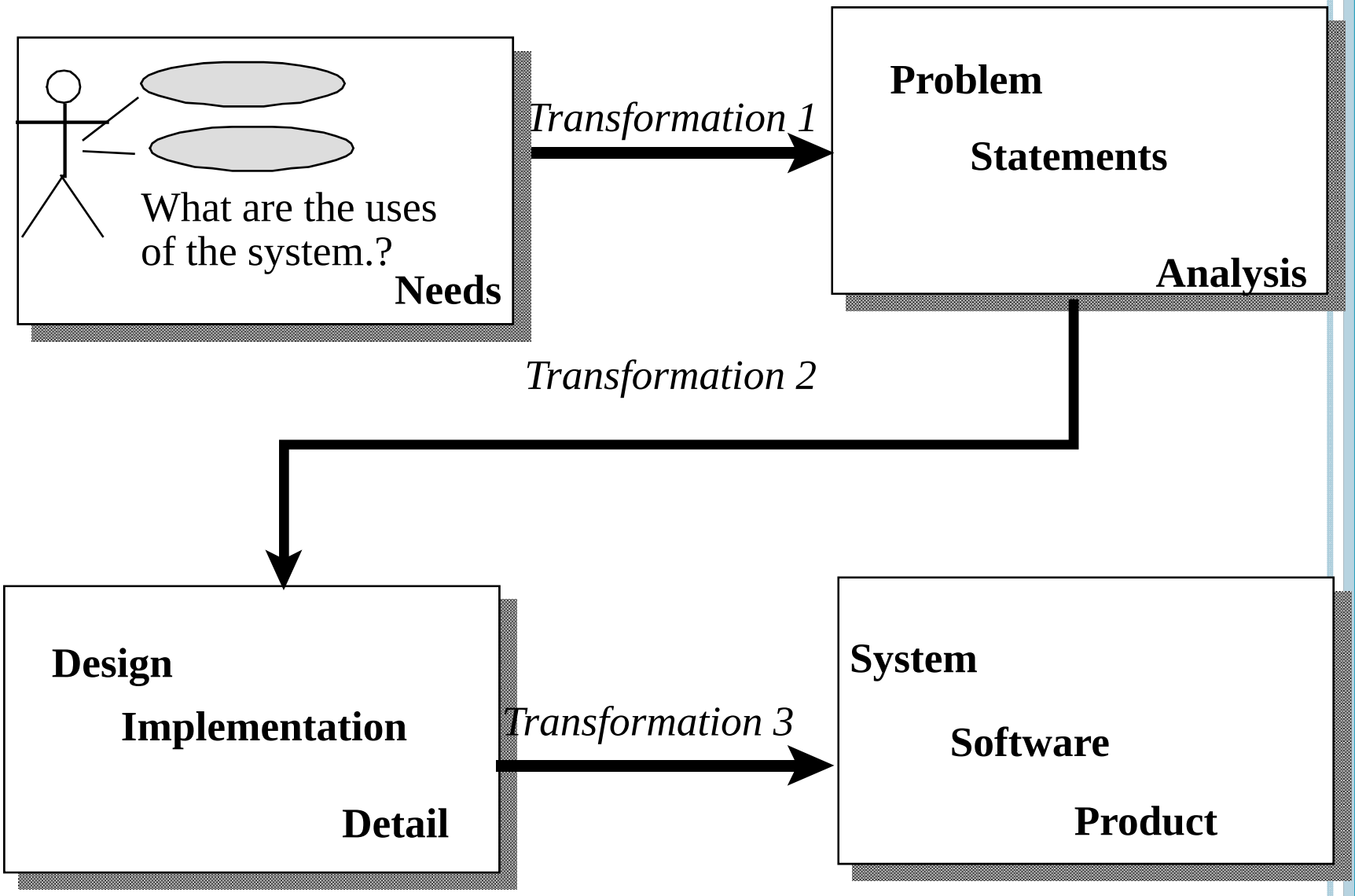
- S/W development can be viewed as a process, which can be divided into interacting phases – subprocesses
- Each subprocess must have the following:
  - A description in terms of how it works
  - Specification of the input required for the process
  - Specification of the output to be produced
- It can also be divided into smaller, interacting subprocesses.
- That can be viewed as a series of transformations, where the output of one transformation becomes input of the next one

SOFTWARE PROCESS

- The essence of the software process can be viewed as a series of transformation, where the output of one transformation becomes the input of the sequent transformation.

- Can be viewed as a process that can be divided into small independent interacting phases called sub-processes
  - A description in terms of how it works
  - Input specification required by the process
  - Specification of the output to be produced

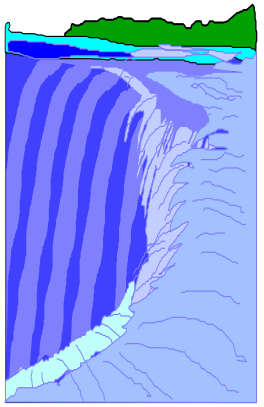
SOFTWARE PROCESS (CON'T)

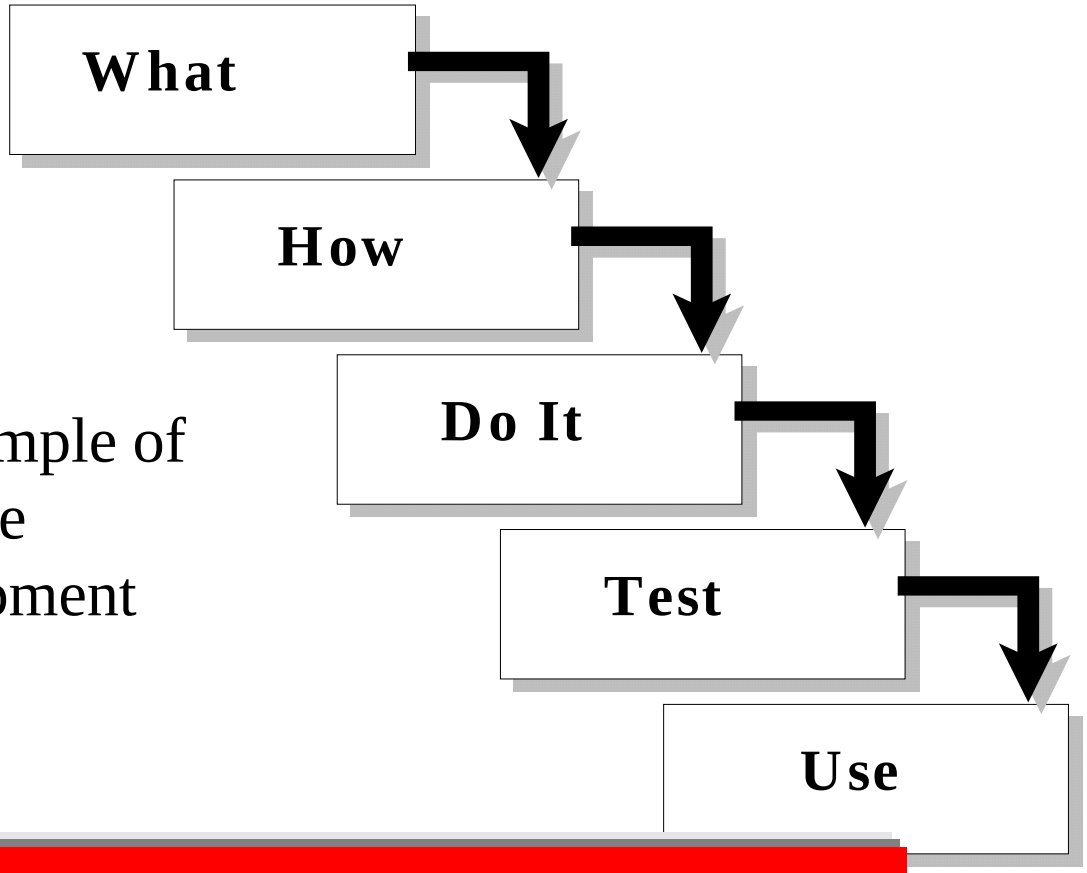


SOFTWARE PROCESS

- Transformation 1 (Analysis) translates Users' needs into system requirements/responsibilities. The user requirements can be have from the way they use the system
- Transformation 2(Design) begins with a problem statement and ends with detailed design that can be transformed into an operational system. Includes the bulk of SW development activities
- Transformation 3(Implementation) System deployment that will satisfy the users needs.
  - Takes into account the procedures, resources - equipment, people etc
  - A software solution

TRADITIONAL WATERFALL APPROACH TO SYSTEMS DEVELOPMENT





An example of software development process

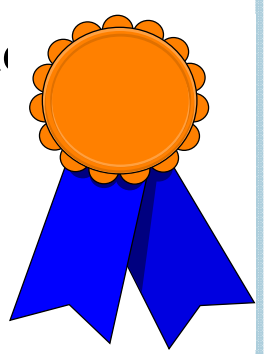
BUT, Real-world problem does not fall into WF Model

WATER FALL MODEL

- Starts with deciding what is to be done (What is the problem?)
- Assumes that the requirements will remain static over the development cycle
- Based on well established engineering principle

SOFTWARE : HIGH-QUALITY

- Must meet user's needs and expectation (User satisfaction)
- Products should attain this with minimal or no defects
- (Improving products (or services) prior to delivery rather than correcting them after delivery)
- Once a system exists, we must test to see it is free of bugs as the ultimate goal of building high quality s/w is user satisfaction
- To achieve high quality software we need to ask following questions:
  - How do we determine the system is ready for delivery?
  - Is it now an operational system that satisfies users' n
  - Is it correct and operating as we though it should?
  - Does it pass an evaluation process?

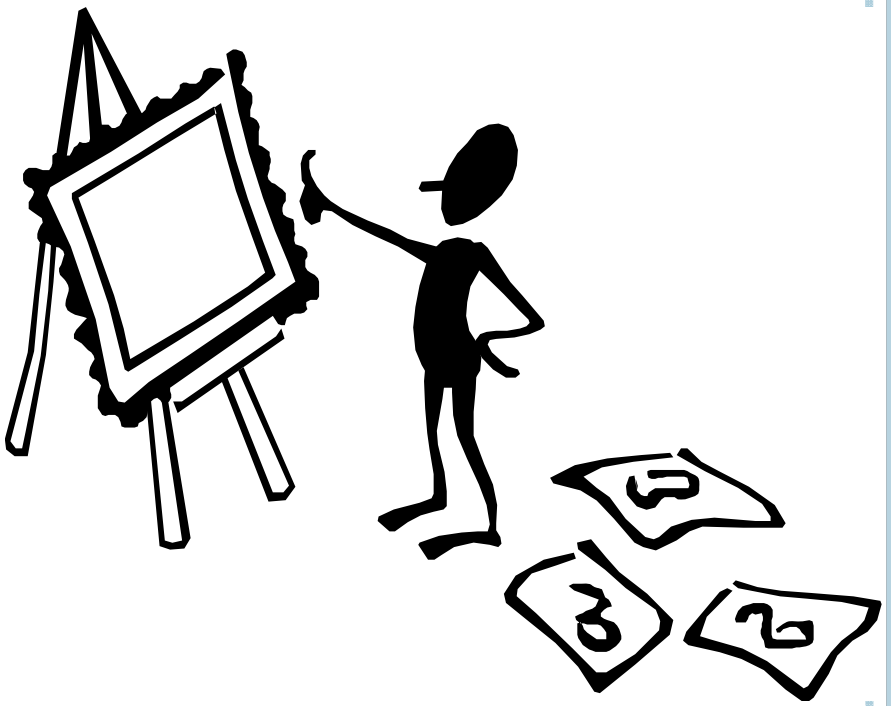


SYSTEM TESTING

- There are two basic approaches
  - We can test a system according to how it has been built.
  - We can test the system with respect to what it should do.

QUALITY MEASURES

- Systems can be evaluated in terms of four quality measures: [BLUM]
  - Correspondence
  - Correctness
  - Verification
  - Validation



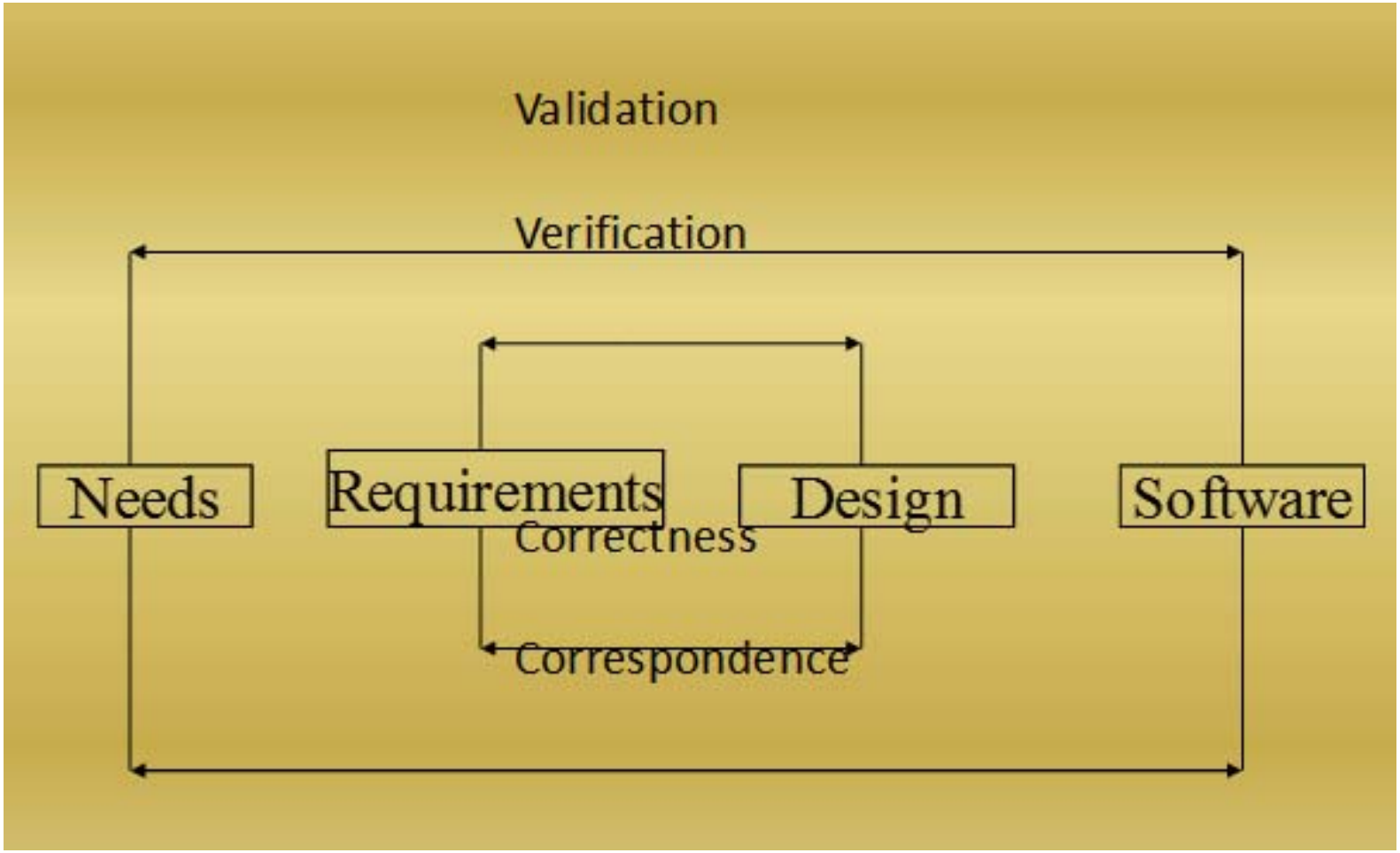
QUALITY MEASURES (CON'T)

- Correspondence – Measures how well the delivered system matches the needs of the operational environment as described in the original requirements statements
- Correctness – Measures the consistency of the product requirements with respect to the design specification [it is objective]

QUALITY MEASURES (CON'T)

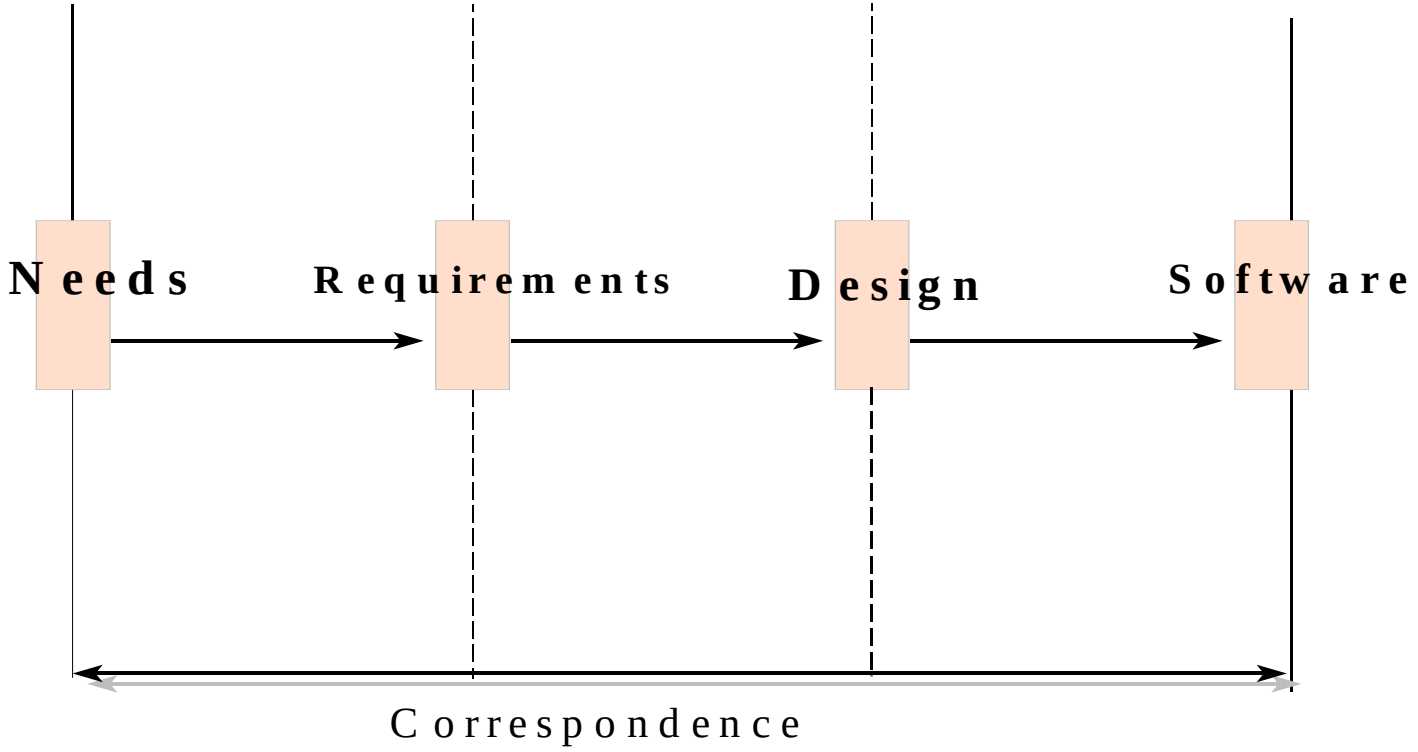
- Verification – It is the task of determining correctness
- Verification : Am I building the product right
- Validation – It is the task of predicting correspondence. Correspondence cannot be predicted until the system is in place[it is subjective]
- Validation : Am I building the right product
- Validation starts as soon as the project begins
- Verification begins only after a specification has been accepted
- Both are independent of each other

QUALITY MEASURES (CON'T)



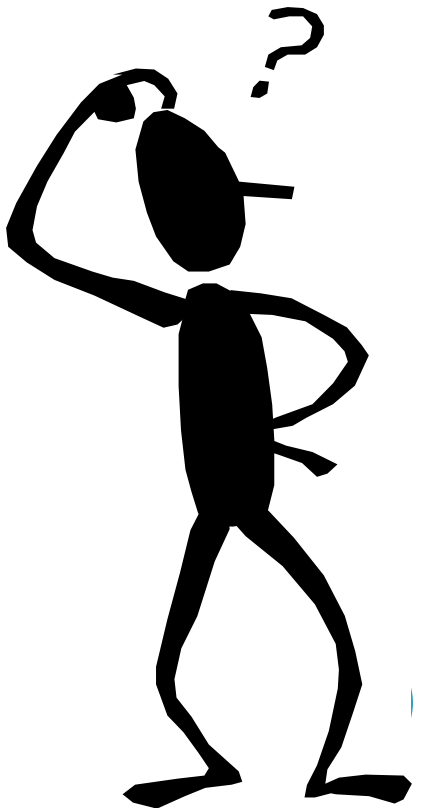
QUALITY MEASURES (CON'T)

- Correspondence measures how well the delivered system corresponds to the needs of the operational environment



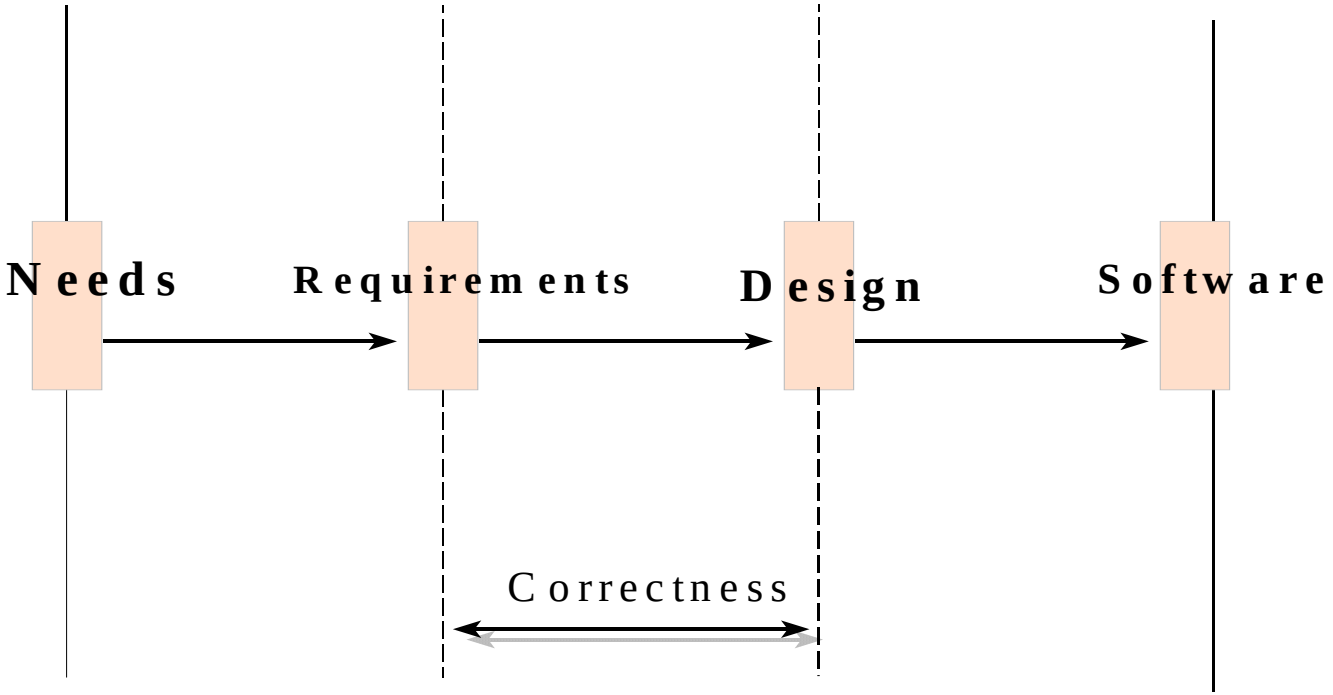
HOW WOULD YOU DETERMINE CORRESPONDENCE?

- It cannot be determined until the system is in place



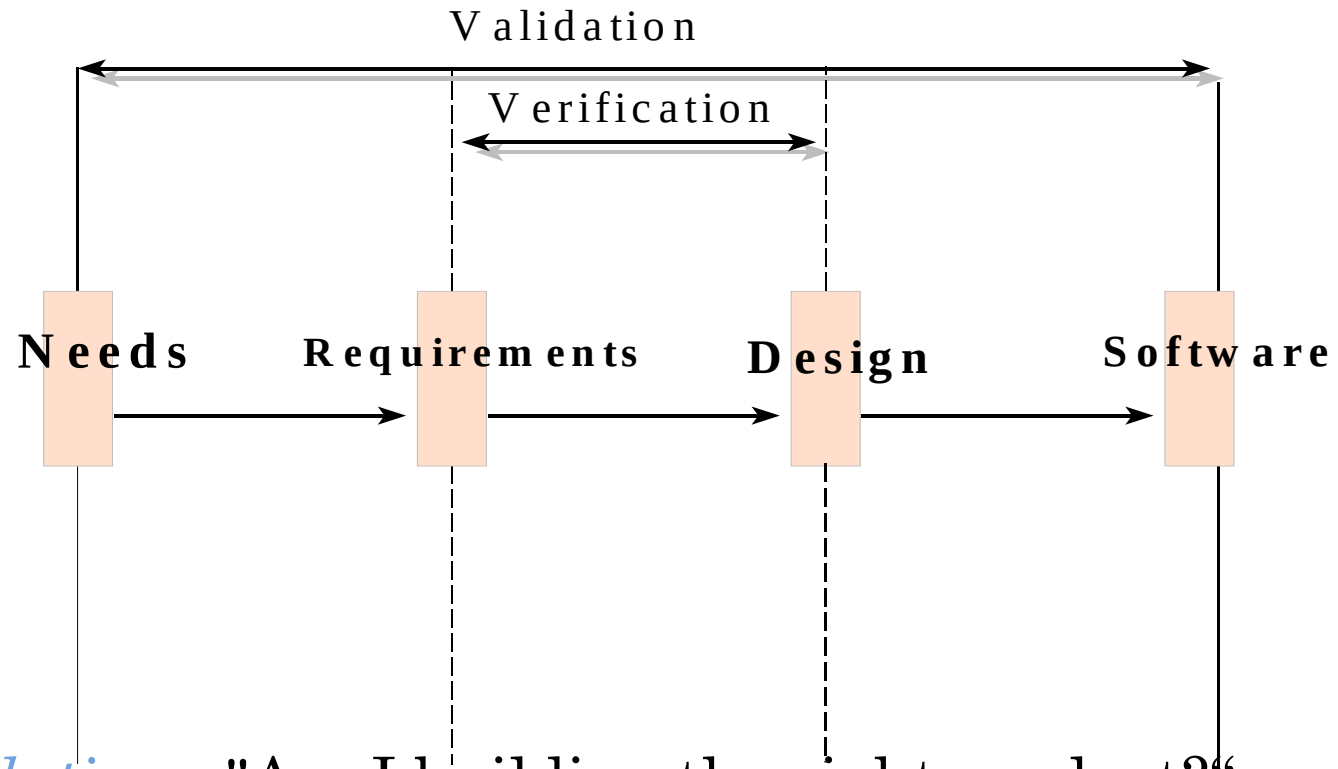
QUALITY MEASURES (CON'T)

- Correctness measures the consistency of the product requirements with respect to the design specification



QUALITY MEASURES (CON'T)

- *Verification* - "Am I building the product right?"
- *Verification* is to predict the correctness.



- *Validation* - "Am I building the right product?"
- *Validation* is to predict the correspondence.

OBJECT-ORIENTED SYSTEMS DEVELOPMENT LIFE CYCLE

- The object oriented software development life cycle (SDLC) consists of three macro processes:
  - Object-oriented analysis
  - Object-oriented design
  - Object oriented implementation

OBJECT-ORIENTED SYSTEMS DEVELOPMENT LIFE CYCLE

- Object-oriented analysis
  - A description of the problem and requirements
  - An investigation of the problem (not the solution)
  - Emphasis on finding and describing the objects or concepts in the problem domain
- **Example - Library Information System**
  - Investigate what the business processes related to its use are?
  - Some of the concept include:
    - Books
    - Library

OBJECT-ORIENTED SYSTEMS DEVELOPMENT LIFE CYCLE

- Object-oriented design
  - Emphasize the logical solution and how it fulfills requirements and constraints
  - Responsibility assignment - allocate tasks and responsibilities to the various software objects in the application
  - Software objects usually collaborate or interact in order to fulfill their responsibilities
  - The illustration of responsibility assignment and object interaction is often expressed with design
    - Class diagrams (show the definition of classes)
    - Collaboration diagrams (show the flow of message between software objects)

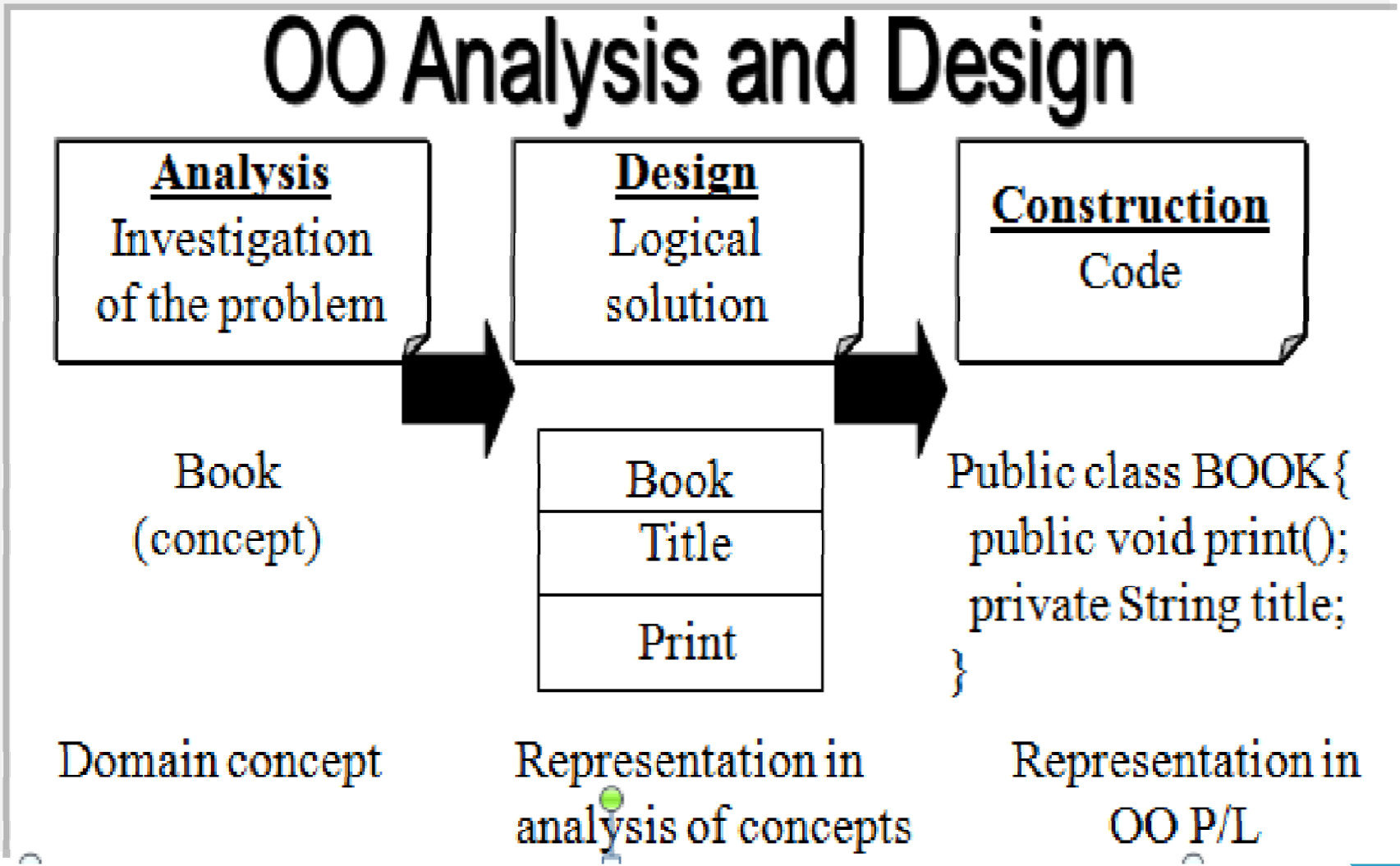
OBJECT-ORIENTED SYSTEMS DEVELOPMENT LIFE CYCLE

- Object-oriented design
  - Define logical software objects that will ultimately be implemented in a (object-oriented) P/L
  - These software objects have
    - Attributes, and
    - Methods
  - **Example - Library Information System**
    - Design should address how exactly will the software capture and record loans
    - Ultimately design can be implemented in Software and Hardware
    - A book software object may have
      - a title attribute, and
      - a print method

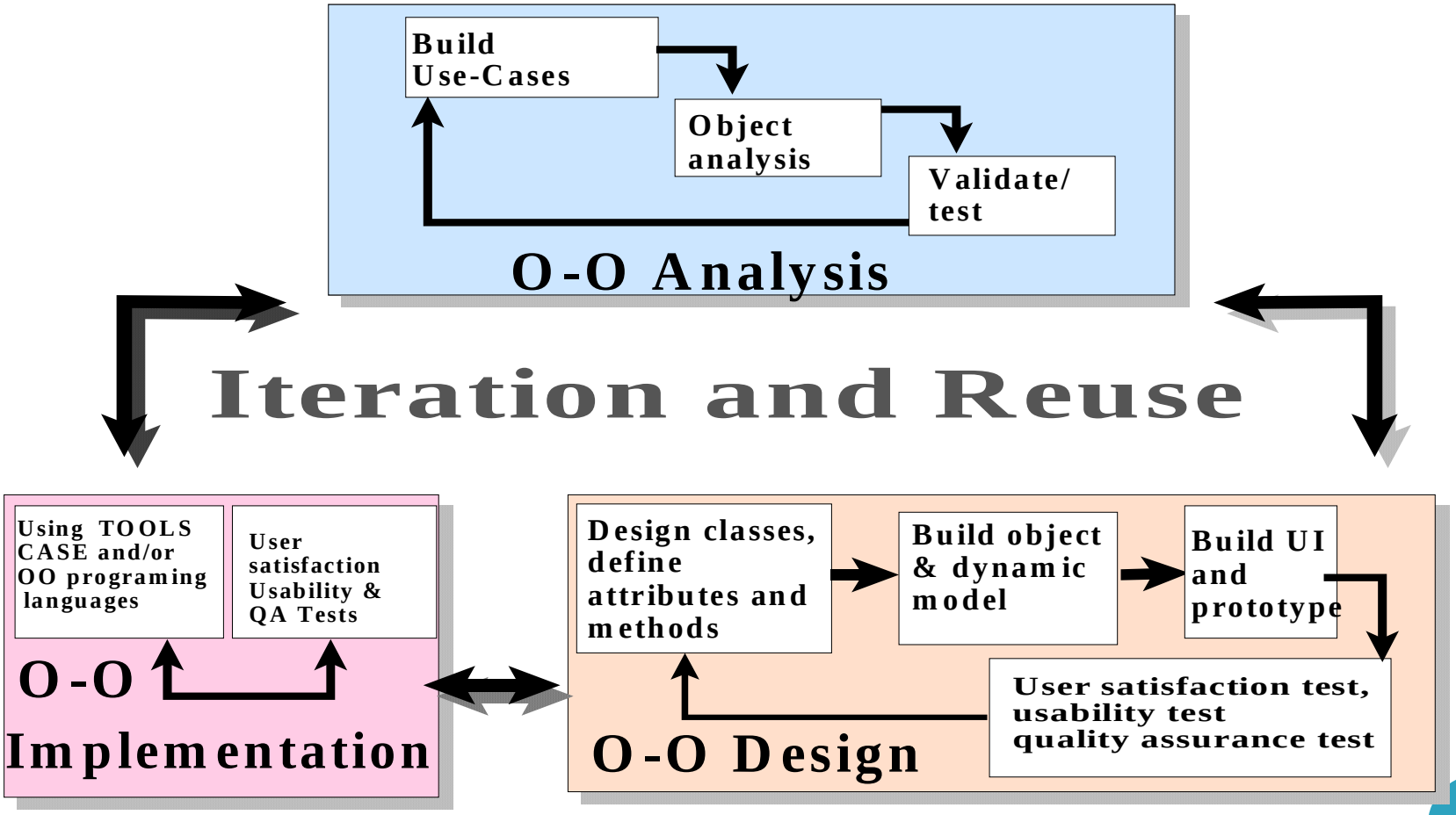
OBJECT-ORIENTED SYSTEMS DEVELOPMENT LIFE CYCLE

- Object-oriented implementation
  - Also referred to as construction, or coding
  - Design components are implemented such as Book class in Java or C++

Object-orientation emphasizes representation of objects



OBJECT-ORIENTED SYSTEMS DEVELOPMENT :USE CASE DRIVEN APPROACH

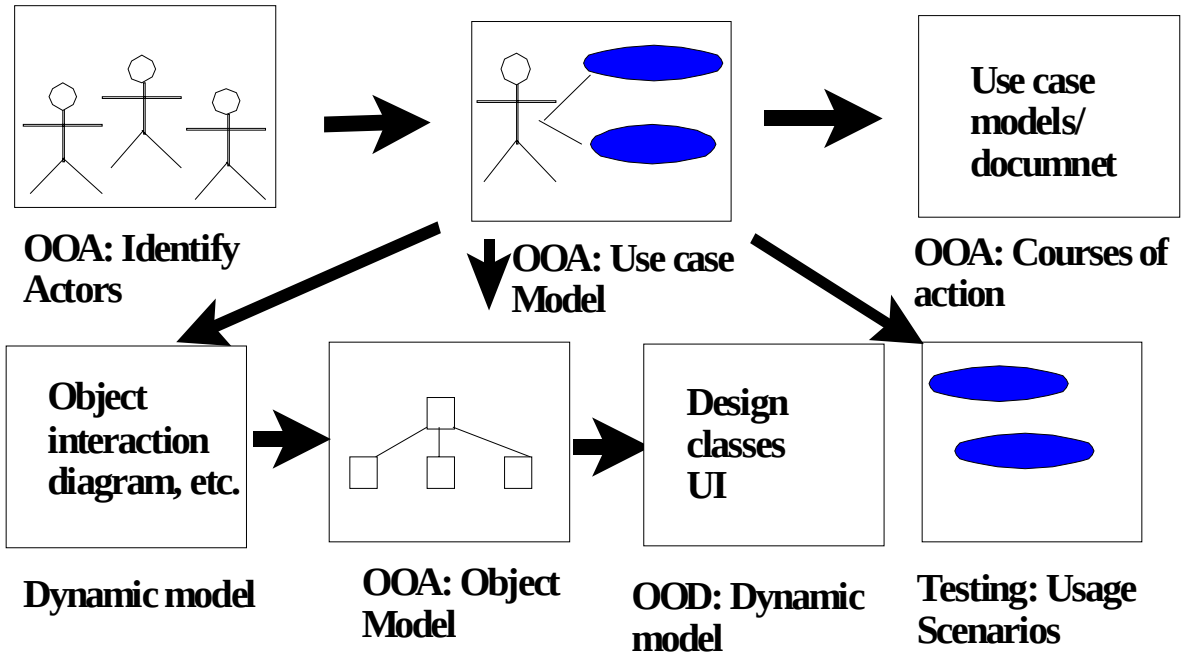


OBJECT-ORIENTED SYSTEMS DEVELOPMENT :USE CASE DRIVEN APPROACH

- Advantages of use case model
  - Can be employed throughout most activities of software development
  - All design decisions can be traced back directly to user requirements
  - by following the life cycle model of Ivar Jacobson.

USE-CASE DRIVEN SYSTEMS DEVELOPMENT

- *Use Case*, is a name for a scenario to describe the user–computer system interaction.
- A use case represents a particular functionality of a system.
- Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers.
- These controllers are known as **actors**.



OBJECT-ORIENTED SYSTEM DEVELOPMENT

- OO system development includes these activities
  - Object oriented analysis – use case driven
  - Object-oriented design
  - Prototyping
  - Component based development
  - Incremental testing.
- It encourages you to view the problem as a system of cooperative objects.

OBJECT-ORIENTED ANALYSIS

- OO analysis concerns with determining the system requirements and identifying classes and their relationships that make up an application
- To understand the system requirements we need to identify users/actors and how they use the system.
- Scenarios are a great way to examining who does what in the interactions among objects and what role they play i.e. their relationship.
- This interaction among objects' roles to achieve a given goal is called collaboration.

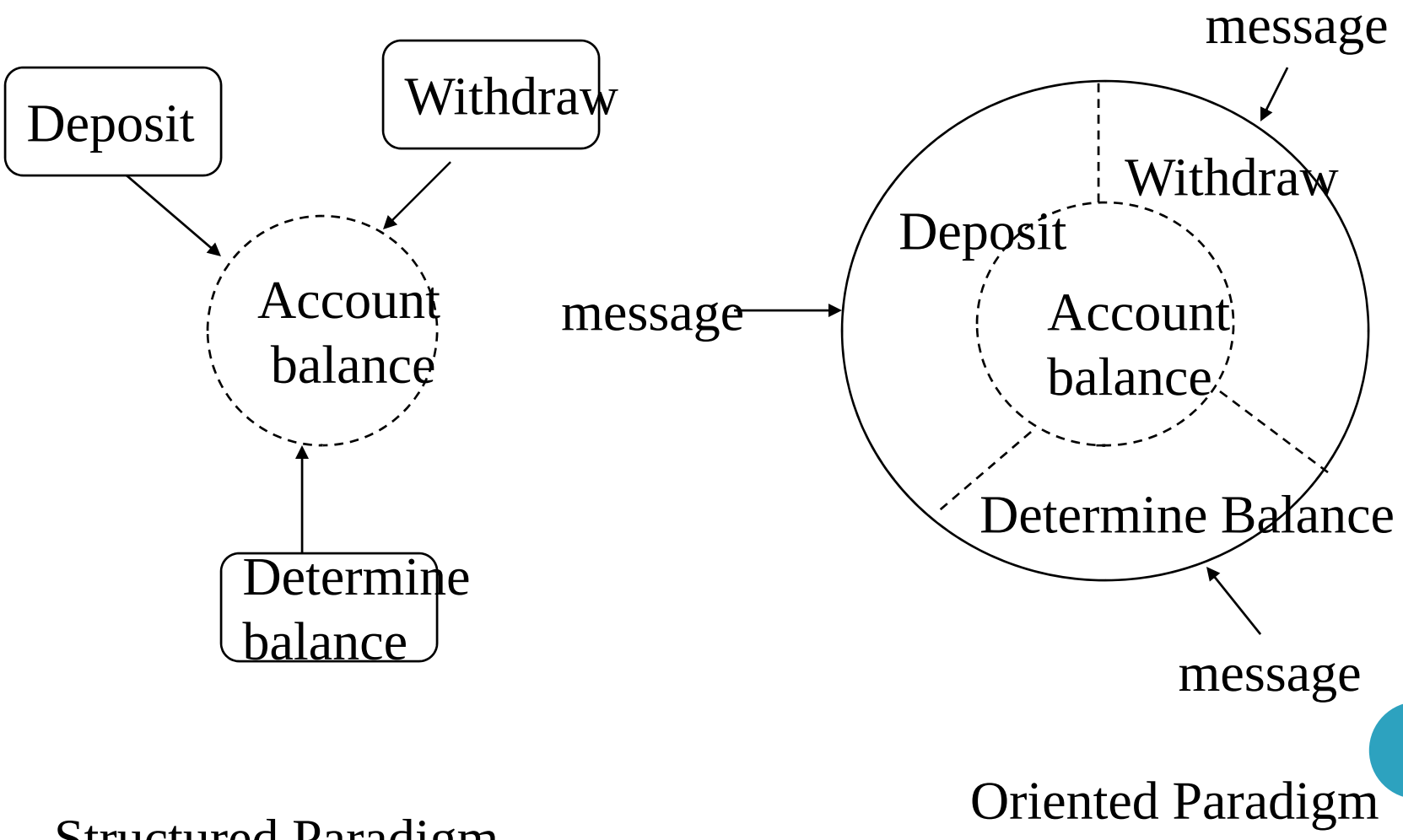
OBJECT-ORIENTED DESIGN

- The goal of *object-oriented design* (OOD) is to design:
  - The classes identified during the analysis phase,
  - The user interface and
  - Data access

OBJECT-ORIENTED DESIGN (CONT..)

- OOD activities include:
  - Design and refine classes
    - Design and refine attributes
    - Design and refine methods
    - Design and refine structures
    - Design and refine associations
  - Design User Interface or View layer classes
  - Design data Access Layer classes

Structured vs OO Example



PROTOTYPING

- A Prototype enables you to fully understand how easy or difficult it will be to implement some of the features of the system
- It can also give users a chance to comment on the usability and usefulness of the design
- It has been said that “A picture may be worth a thousand words but a prototype is worth a thousand pictures”

TYPES OF PROTOTYPES

- A *horizontal prototype* is a simulation of the interface
  - It has the entire user interface that will be in full-featured system. But contains no functionality
- A *vertical prototype* is a subset of the system features with complete functionality
  - So few implemented functions can be tested in great depth

TYPES OF PROTOTYPES (CON'T)

- An *analysis prototype* is an aid for exploring the problem domain- used to inform the user and demonstrate the proof of a concept
  - It cant be used as basis for development, and discarded when served it purpose

TYPES OF PROTOTYPES (CON'T)

- A *domain prototype* is an aid for the incremental development of the ultimate software solution
  - Used for staged delivery of the subsystems to the users
  - It demonstrates the feasibility of the implementation and eventually will evolve into a deliverable product
- Time required to produce a prototype can be from a few days to some weeks- it depends on the type and function of prototype

CONT..

- Prototyping should involve representation from all user groups that will be affected by the project
- Especially the end users and management members to ascertain that the general structure of the prototype meets the requirements established for the overall design

PURPOSE OF THE REVIEW (PROTOTYPE)

- First, to demonstrate that the prototype has been developed according to the specification and that final specification and that the final specification is appropriate
- Second, to collect information about the errors or other problems in the system, such as user interface problems that need to be addressed in the intermediate prototype stage
- To give management and everyone connected with the project the first (or it could be second or more..) glimpse of what the technology can provide

CONT..

- Prototype is a useful exercise at any stage of development
- It should be done in parallel with the preparation of the functional specification
- With key features specified, prototyping those features usually results in
  - modifications to the specifications and
  - even reveal additional features or problems that were not obvious until the prototype was built

COMPONENT-BASED DEVELOPMENT (CBD)

- CBD makes many products available to marketplace that otherwise would be prohibitively expensive
- s/w components are built and tested in-house, using a wide range of technologies like CASE (Computer-aided s/w engineering) tools
- CBD is an industrialized approach: With focus shifted from software programming to composing assembly of pre-built, pre-tested, reusable software components that operate with each other

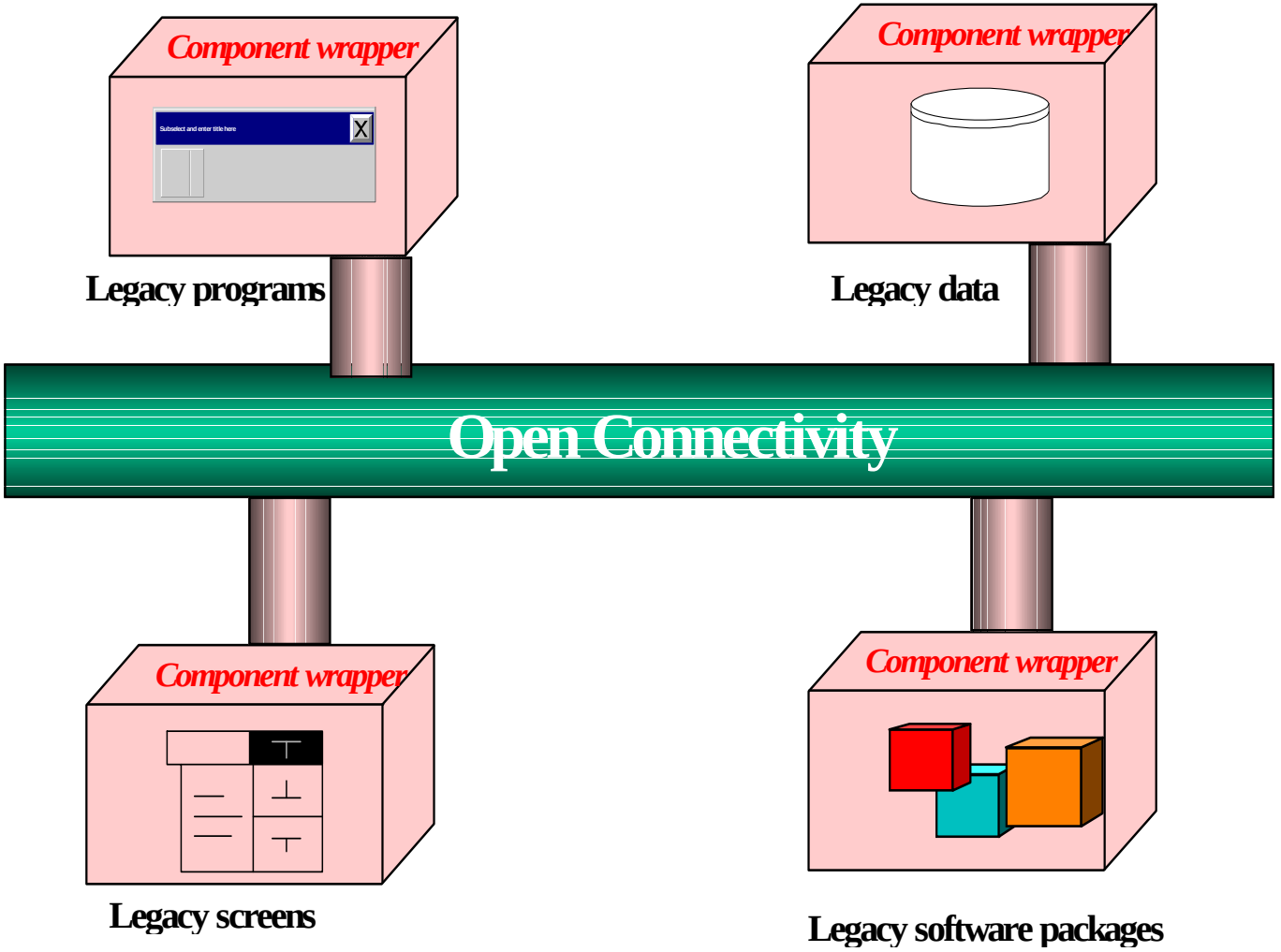
TWO BASIC IDEAS

- First, application development can be improved if applications can be assembled quickly from prefabricated software components
- Second, an increasingly large collection of interpretable software components could be made available to developers in both general and specialist catalogs
- The software components are the functional units of a program, building blocks offering a collection of reusable services

COMPONENT-BASED DEVELOPMENT (CBD)

- The CBD developer can assemble components to construct a complete software system using a well-defined software architecture.
- Components themselves can be built from other components
- The software components are the functional units of a program, building blocks offering a collection of reusable services

COMPONENT-BASED DEVELOPMENT (CBD) CON'T)



RAPID APPLICATION DEVELOPMENT (RAD)

- RAD is a set of tools and techniques that can be used to build an application faster than typically possible with traditional methods
- It can be used in conjunction with software prototyping- here the developer sacrifices the quality of the product for early delivery
- It concerned with reducing “time to market”, not exclusively the software development time

RAPID APPLICATION DEVELOPMENT (RAD) (CON'T)

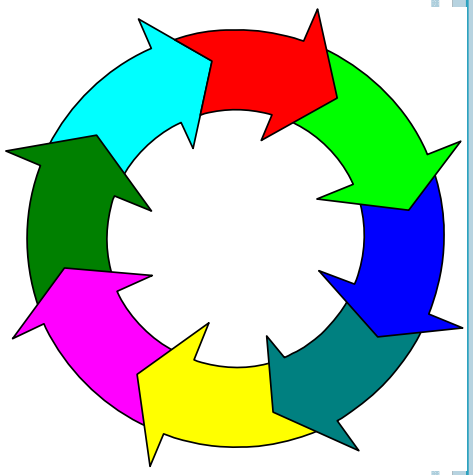
- An incremental alternative to traditional Waterfall Model
- RAD does not replace SDLC but complements it, since it focuses more on process description and can be combined perfectly with the object-oriented approach
- The task of RAD is to build the application quickly and incrementally implement the design and user requirements through the tools such as Delphi, VisualAge, VB and PowerBuilder

RAPID APPLICATION DEVELOPMENT (RAD) (CON'T)

- Prototyping and RAD does not replace OO-S/W development
- Instead, in RAD application, you go through those stages in rapid fashion, completing a little more in the next iteration of the prototype

INCREMENTAL TESTING

- Software development and all of its activities including testing are an iterative process
- If you wait until after development to test an application for bugs and performance, you could be wasting thousands of dollars and hours of time



REUSABILITY

- A major benefit of object-oriented systems development is reusability, and this is the most difficult promise to deliver on
- The potential benefits are: increased reliability, reduced time and cost of development, and improved quality
- To evaluate existing s/w components of reusability ask following questions:
  - Has my problem already been solved?
  - Has my problem partially solved?
  - What has been done before to solve the similar problem to his one?

REUSE STRATEGY

- Information hiding (encapsulation).
- Conformance to naming standards.
- Creation and administration of an object repository.

REUSE STRATEGY (CON'T)

- Encouragement by strategic management of reuse as opposed to constant redevelopment
- Establishing targets for a percentage of the objects in the project to be reused (i.e., 50 percent reuse of objects)

SUMMARY

- The essence of the software process is the transformation of users' needs into a software solution
- The O-O SDLC is an iterative process and is divided into analysis, design, prototyping/ implementation, and testing