

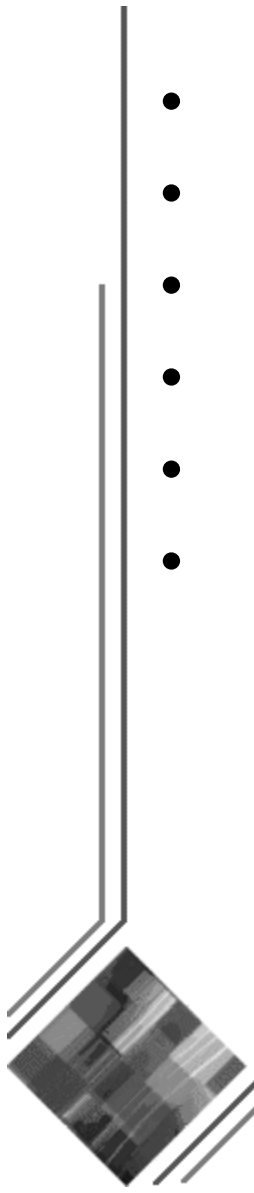


Object-Oriented Software Engineering

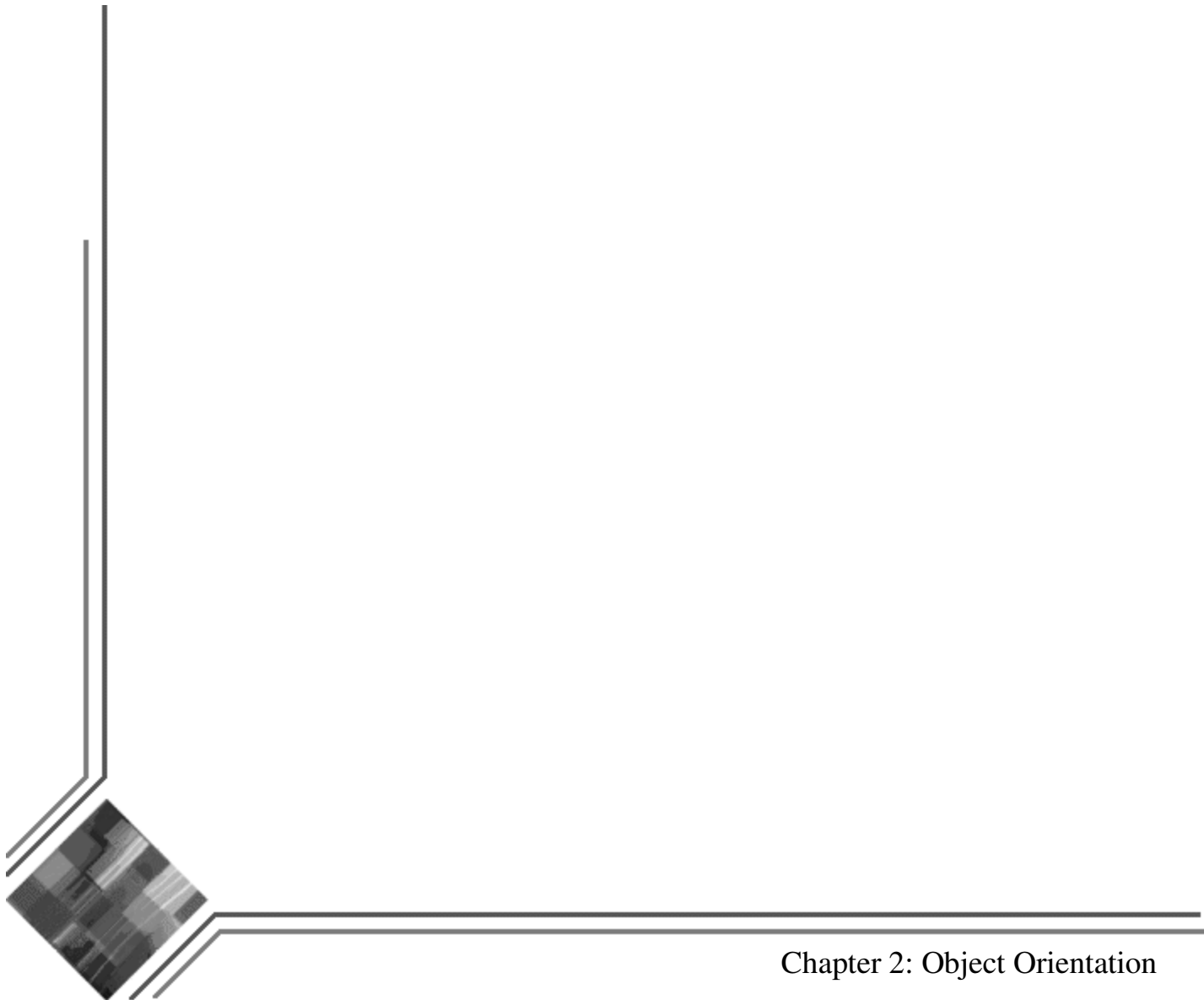
Chapter 2: Object Orientation (OO)

Table of Contents

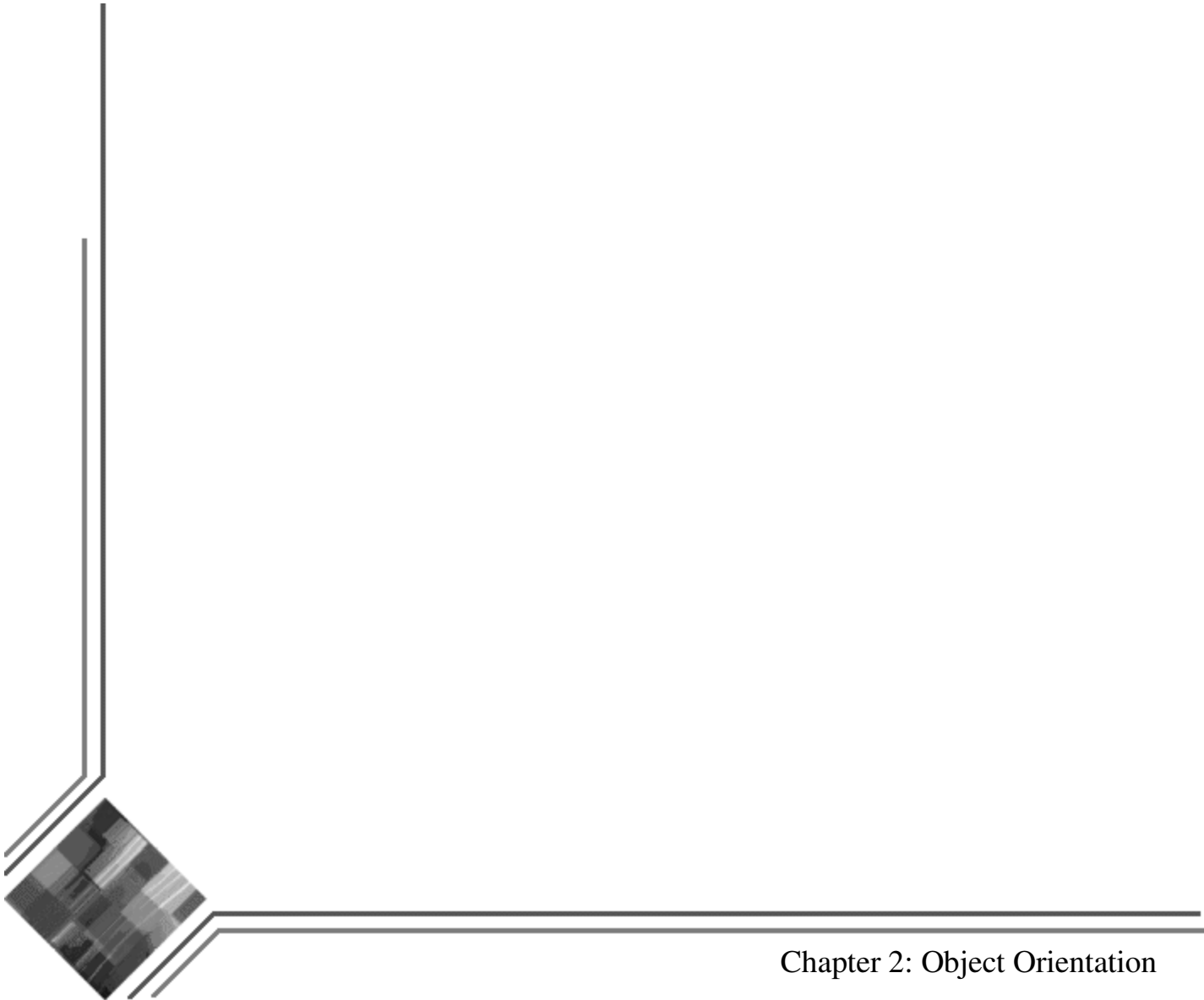
- **Software Process Cycle**
- **Methodology**
- **Conventional Approach**
- **The OO Paradigm**
- **The View of Two Paradigms**
- **The OO Process Model**



Chapter's Directory...



Chapter's Directory



Software Process Cycle

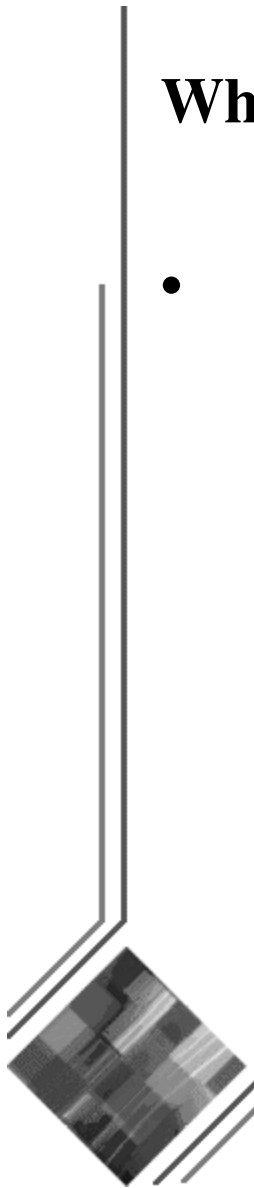
Software Development Process Cycle:

- **Required techniques to be organised into appropriate developmental life cycle**
- **Process of software development -> the tasks and how they are carried out and how they are organised**

Methodology...

What is Methodology?

- **Methodology: set of general principles that guide a practitioner/manager to the choice of the particular method for their project**



Methodology

Why do we use Methodology?

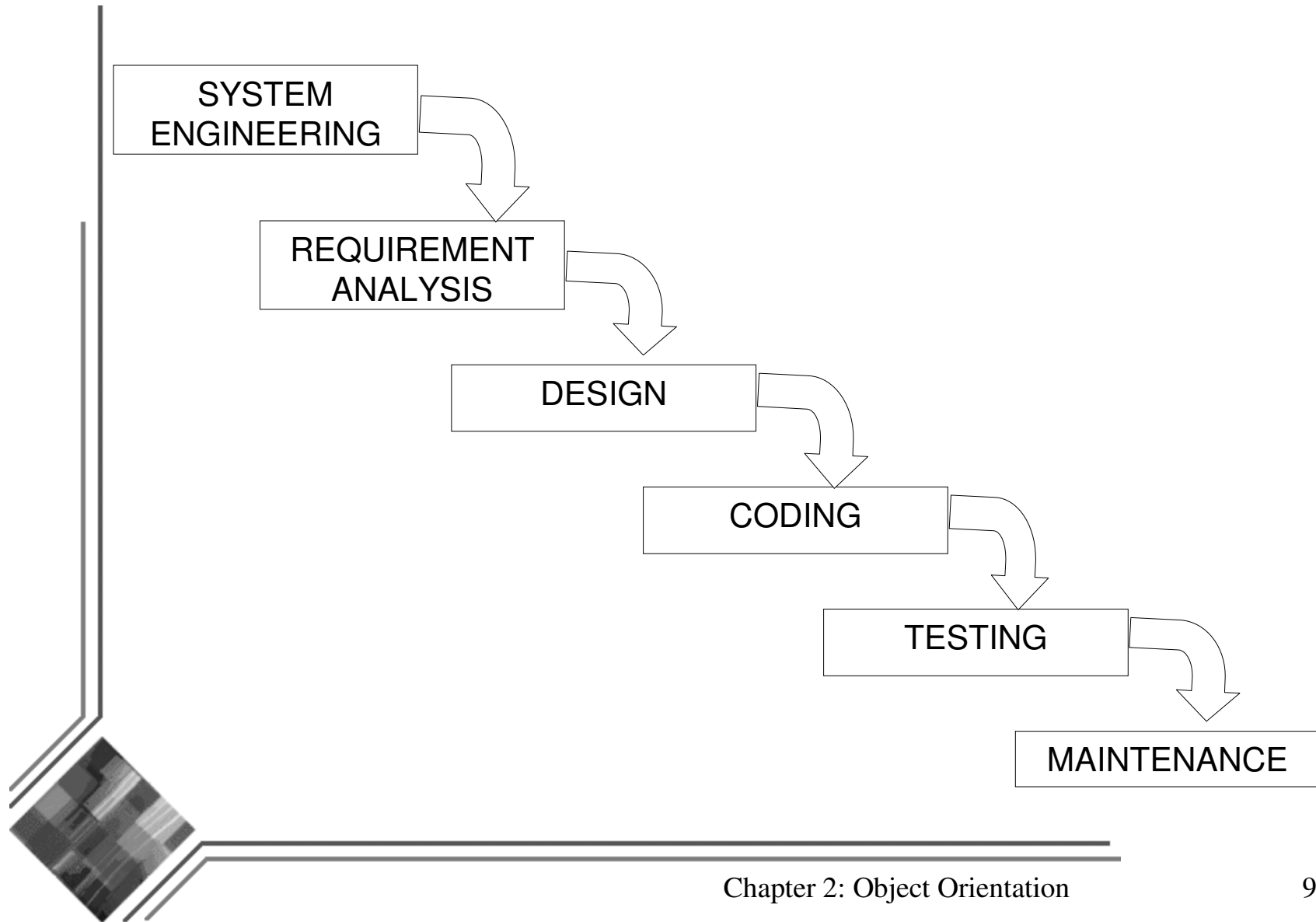
- **Helps to produce better quality products, in terms of documentation standards, acceptability to the user, maintainability and consistency of software**
- **Help to ensure the user requirements are met**
- **Helps the project manager, by giving better control of project execution and reduction in overall development costs**
- **Promote communication between project members**
- **Encourage the transmission of know-how throughout the organization through standardisation process and documentation**

Conventional Approach...

Traditional Waterfall Model

- **This model describes a development method that is linear and sequential**
- **Each phase has distinct goals to achieve**
- **The process moves to next phase upon completion of previous phase without turning back**
- **Straightforward, simple to understand and use**

Conventional Approach...



Conventional Approach...

The Steps:

- ★ Identify system requirements and analyze them
- ★ Break the system into pieces (Architectural design)
- ★ Design each piece (Detailed design)
- ★ Code the system components and test them individually (Coding, debugging, and unit testing)
- ★ Integrate the pieces and test the system (System Testing)
- ★ Deploy the system and operate it

Conventional Approach...

System engineering

- High level specification, define major elements: human, software, hardware and how they interact

Requirement analysis

- Define user requirements
- Use fact finding techniques and document them (interview, DFD..etc.)
- The specification contains:
 - Detailed description of all the tasks constraints, design/ implementation directives AND
 - Maintenance support for system, training of clients

Conventional Approach...

Design

- Determines how to construct a system that delivers the requirements
- Specification of software architecture

Coding

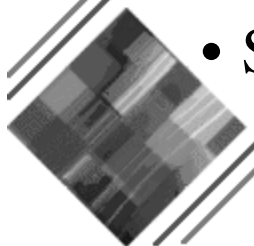
- Translation of design into program code

Testing

- Ensures that system meets requirements, may consist of several levels of testing

Maintenance

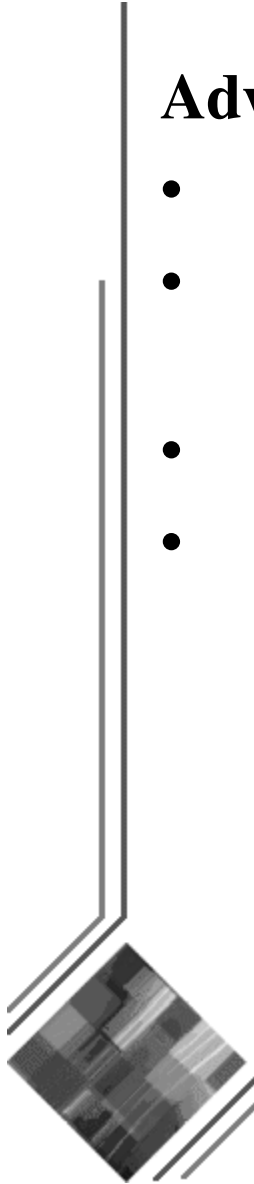
- System is subject to change



Conventional Approach...

Advantages:

- **Easy to understand and implement**
- **Provides template in which analysis, design, coding, testing and maintenance can be placed**
- **Allow departmentalisation and managerial control**
- **Each phase is strict in order without overlapping**



Conventional Approach...

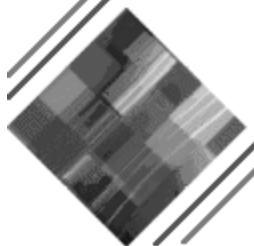
Drawbacks:

- **The sequential flow is rarely followed in real project**
- **Difficult for customer to state all requirements at one time**
- **Delivery of working product takes long time**
- **Problems are not discovered until testing**
- **Unresponsive to changes**
- **Maintenance cost – 70% of system costs**
- **Repairing problem further along the lifecycle becomes progressively more expensive**

Conventional Approach...

Software Prototyping

- **Prototype: system that is partially complete that is built quickly to explore some aspects of the system requirements**
- **Construct with various objectives**
- **To get feedback from users – to understand the system better and improve the prototype**
- **Model is then discard and actual system is engineered**



Conventional Approach...

Selecting the Prototyping Approach:

- **The prototyping paradigm can be either close-ended or open-ended**
- **The close-ended approach is often called throwaway prototyping**
 - Using this approach, a prototype serves as a rough demonstration of requirements
 - It is then discarded, and the software is engineered using a different paradigm
- **The open-ended approach is often called evolutionary prototyping**
 - Uses the prototype as the first part of an analysis activity that will be continued into design and construction

Conventional Approach...

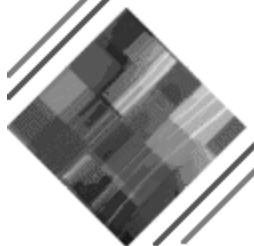
Selecting the Prototyping Approach:

- **What do I look for to determine whether or not prototyping is a viable approach?**
- **Before a close-ended or open-ended approach can be chosen, it is necessary to determine whether the system to be built is amenable to prototyping**
- **Class discussion**

Conventional Approach...

Advantages:

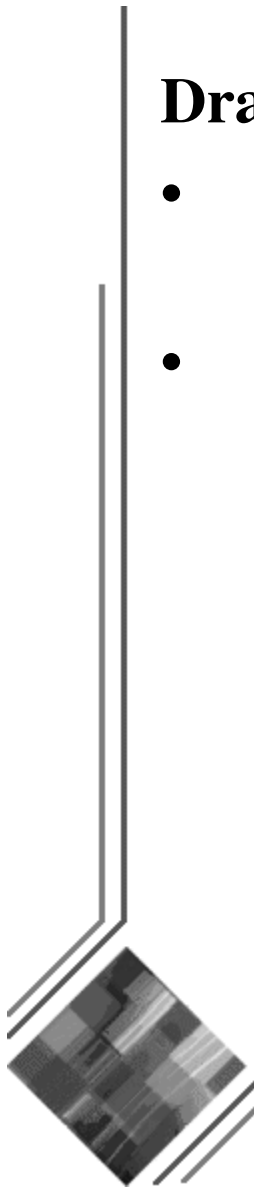
- **Users get a feel for actual system – refine the requirements**
- **Require small teams**
- **Serve as mechanism to identify software requirements**
- **Identification of difficulties in the interface**
- **Feasibility and usefulness of system can be tested**



Conventional Approach...

Drawbacks:

- **Users may perceive the prototype as part of final system**
- **Requires significant user involvement**
 - Difficulties in managing the system life cycle



The OO Paradigm...

Today, the OO paradigm encompasses a complete view of SE

Edward Berard (1993) states:

- The benefits of OO technology are enhanced if it is addressed early-on and throughout the SE process. ... Merely employing OOP will not yield the best results....

Berard, E.V., *Essays on Object-Oriented Software Engineering*, Addison-Wesley, 1993.

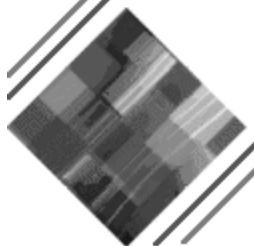
The OO Paradigm...

OO systems make use of abstraction in order to help make software less complex



An abstraction is something that relieves you from having to deal with details

OO systems combine procedural abstraction with data abstraction



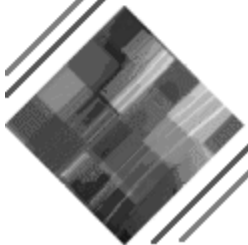
The OO Paradigm...

Procedural paradigm:

- Software is organised around the notion of *procedures*
- *Procedural abstraction*
 - Works as long as the data are simple
- *Adding data abstractions*
 - Groups together the pieces of data that describe some entity
 - Helps reduce the system's complexity.
 - Such as *Records* and *structures*

Object-oriented paradigm:

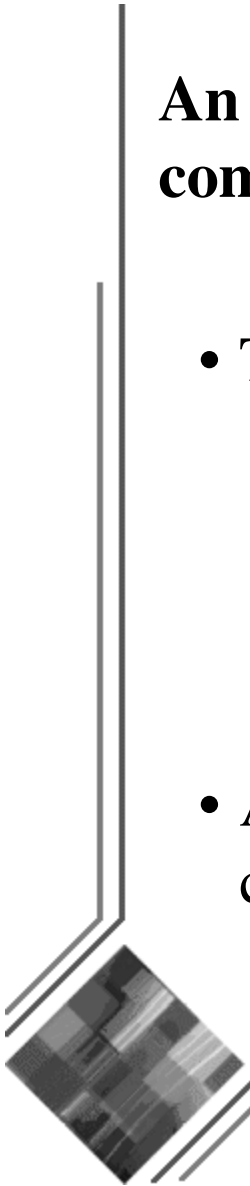
- Organising procedural abstractions in the context of data abstractions



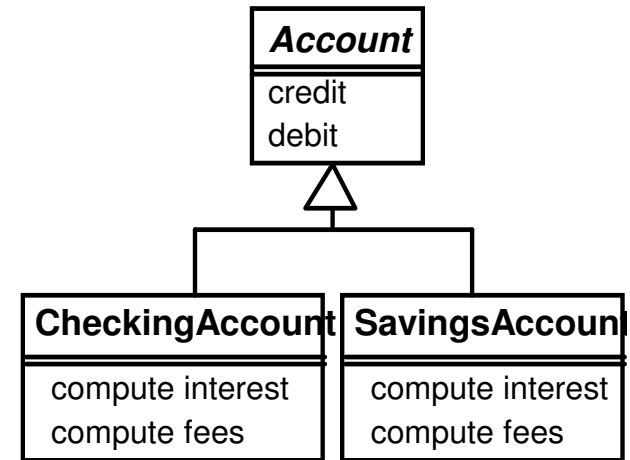
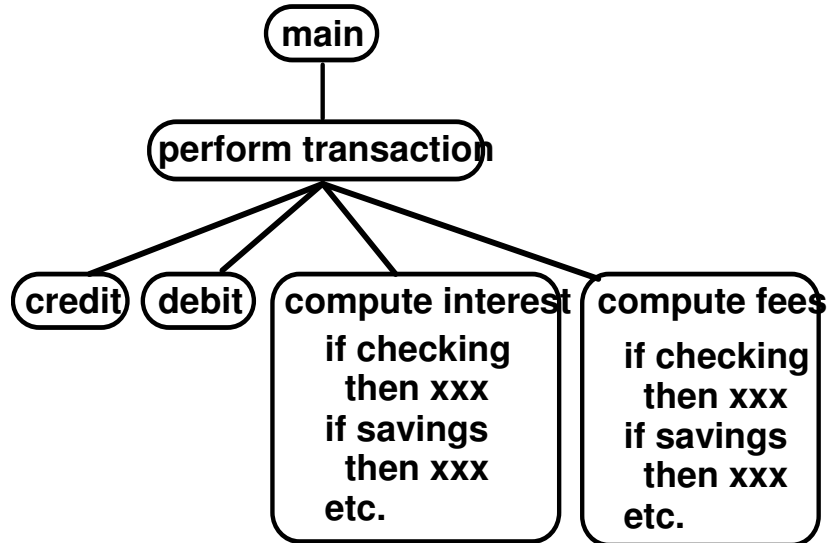
The OO Paradigm...

An approach to the solution of problems in which all computations are performed in the context of objects.

- The objects are instances of classes, which:
 - are data abstractions
 - contain procedural abstractions that operation on the objects
- A running program can be seen as a collection of objects collaborating to perform a given task



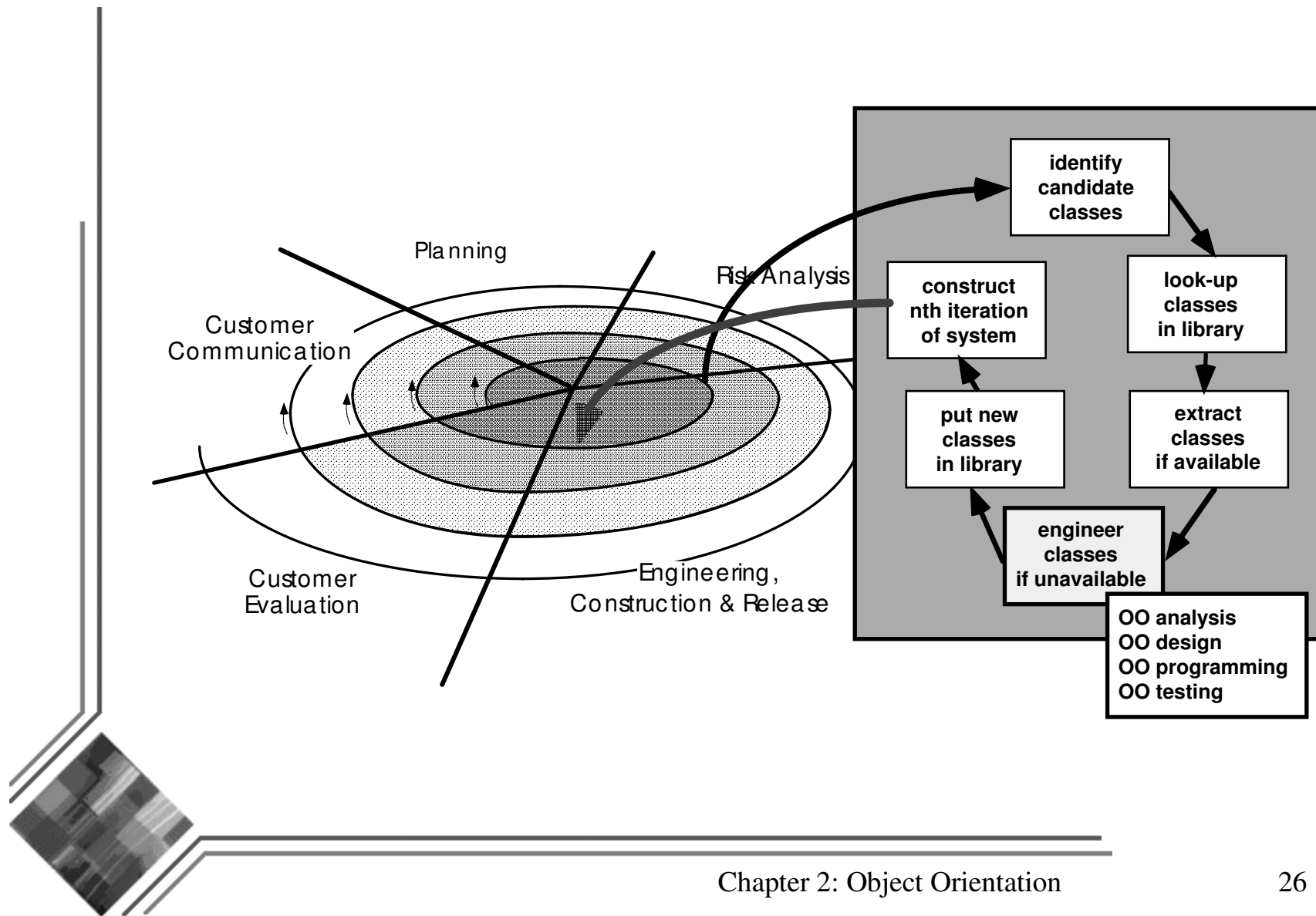
A View of the Two Paradigms



The OO Process Model...

- **There are various process models for SE**
- **Although any one of these models could be adapted for the use with OO, the best choice would recognize that OO systems tend to evolve over time**
- **Therefore, an evolutionary process model, coupled with an approach that encourages component assembly (reuse), is the best paradigm for OOSE**
- **The next figure illustrates the component-based development process model has been tailored for OOSE**

The OO Process Model...



The OO Process Model...

- **The OO process moves through an evolutionary spiral that starts with customer communication**
- **It is here that the problem domain is defined and that basic problem classes are identified**
- **Planning and risk analysis establish a foundation for OO project plan**
- **The technical work associated with OOSE follows the iterative path**
- **OOSE emphasizes reuse (Assignment 1)**
- **OO view demands an evolutionary approach to SE**
- **Suitable for large and complex systems**

The OO Process Model...

Advantages:

Productivity

- OO methodology can increase productivity of project team, reducing project schedule.
- Intrinsic power of OO programming languages
- Reuse of classes and objects

Rapid development

- Reusability
- prototyping

Quality

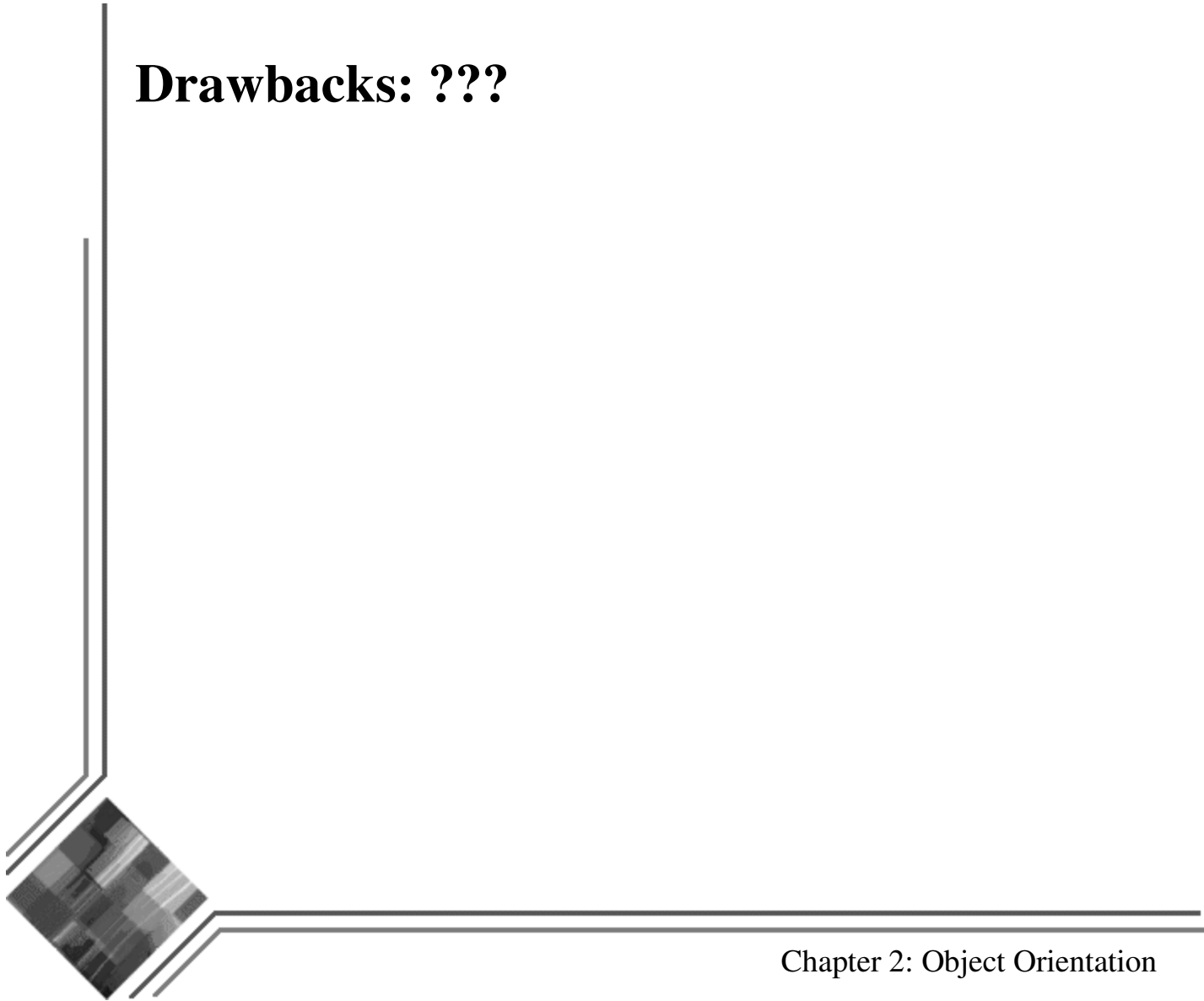
- Low defects
- Reuse, encapsulation

Maintainability

- Greater in OO than conventional architectures

The OO Process Model

Drawbacks: ???



Summary

At the completion of this chapter, you have learnt:

- Conventional approach vs. OO approach in terms of:
 - **Differences**
 - **Models**
 - **Advantages**
 - **Drawbacks**

Review Question

Assume you are future methodologist, do you think the OO methodology is *STILL* sufficient to be used to cater for the uncertainty about technology/SE process/requirements, etc? Why?

Justify with supportive reasons.