Introduction

Pressman chap. 1 +2.

Larman Chap. 1

Software engineering

- ✧The economies of ALL developed nations are dependent on software.
- ✧More and more systems are software controlled
- ✧Software engineering is concerned with theories, methods and tools for professional software development.
- ✧Expenditure on software represents a significant fraction of GNP in all developed countries.

Software costs

- ✧Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- ✧Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- ✧Software engineering is concerned with cost-effective software development.

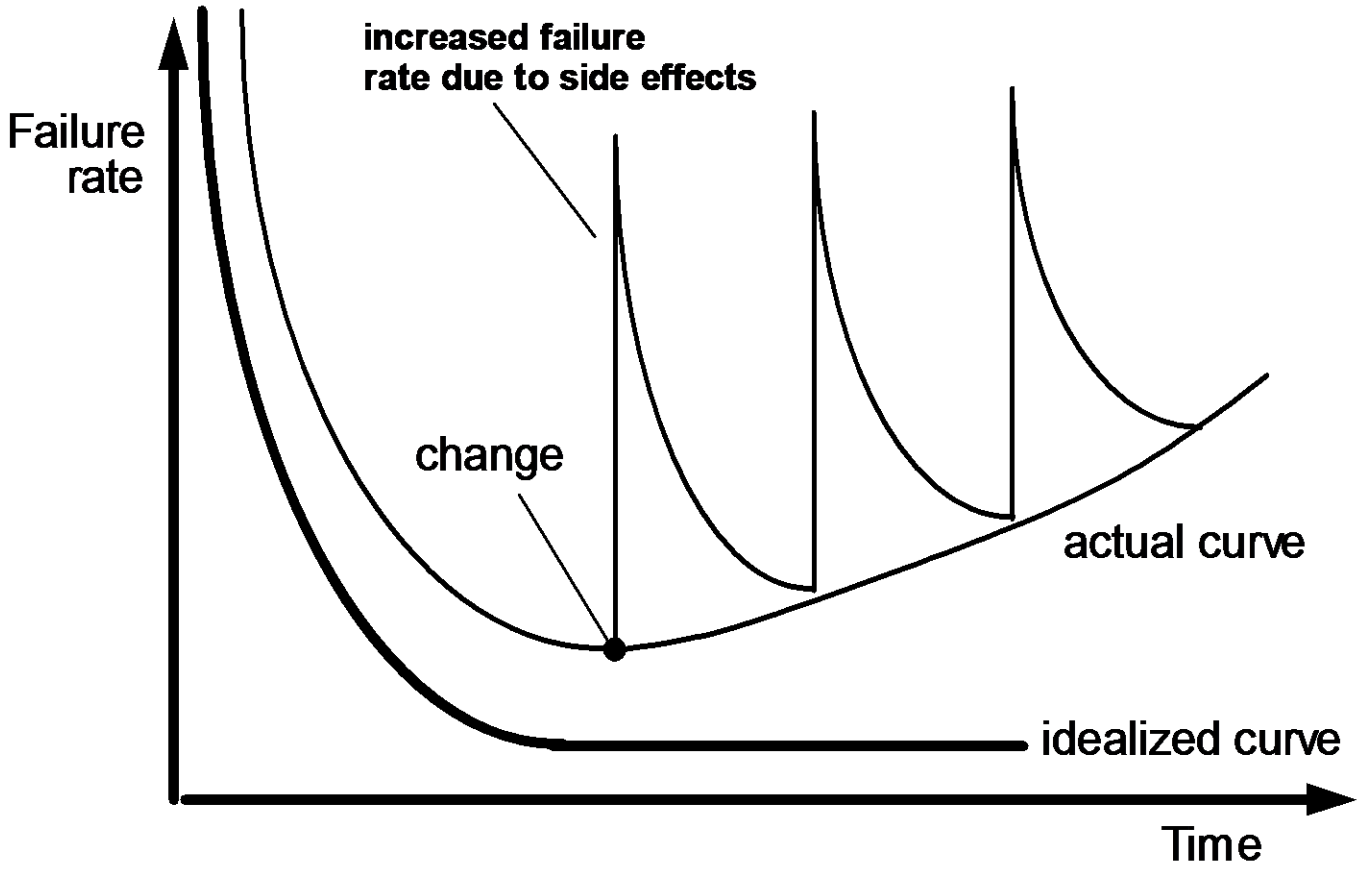
Software products

- ✧Generic products
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
- ✧Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.

Product specification

- ✧Generic products
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
- ✧Customized products
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

Failure curves for software

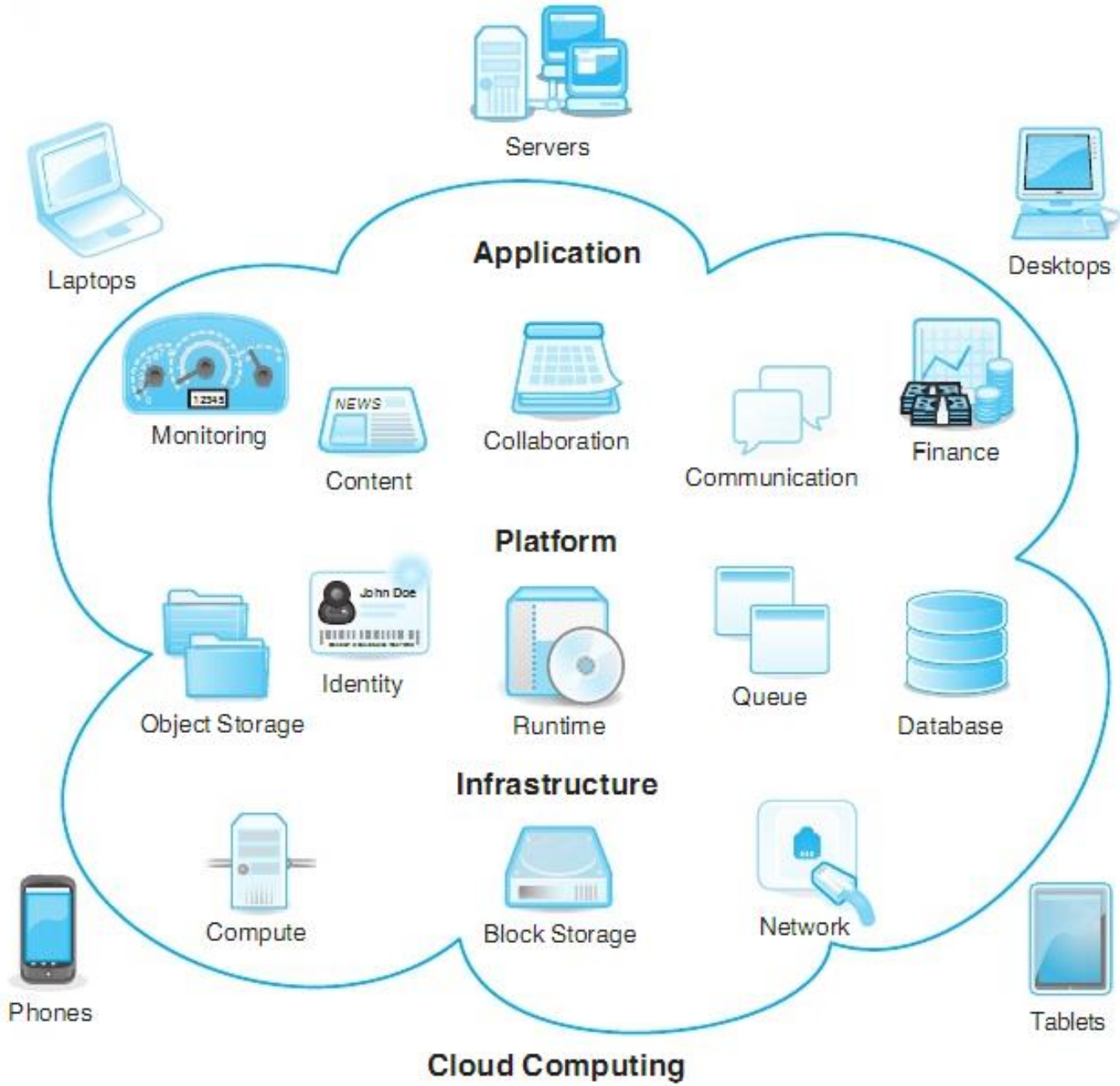


Software Applications

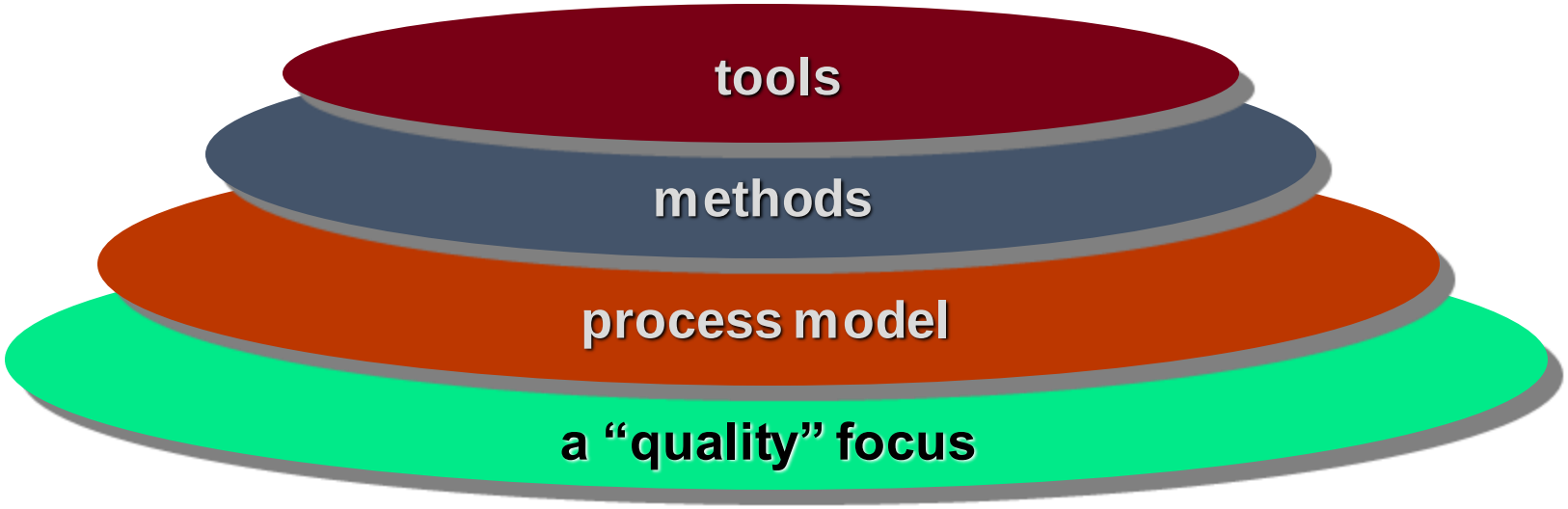
- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

Legacy Software

- ***Why must it change?***
  - software must be adapted to meet the needs of new computing environments or technology.
  - software must be enhanced to implement new business requirements.
  - software must be extended to make it interoperable with other more modern systems or databases.


Servers
Laptops
Desktops
Application
Monitoring
Content
Collaboration
Communication
Finance
Platform
Object Storage
Identity
Runtime
Queue
Database
Infrastructure
Compute
Block Storage
Network
Phones
Tablets
Cloud Computing

A Layered Technology



Software Engineering

Framework Activities – The process framework

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

Umbrella Activities

- Software engineering process frame work activities are complemented by a number of *umbrella activities.* In general, umbrella activities are applied throughout a software project and help a software team manage and control progress , quality, change and risk .

- Software project management
- Formal technical reviews
- Software quality assurance
  - Defines and conducts the activities required to ensure software quality.
- Software configuration management
  - Manages the effects of change throughout the software process .
- Work product preparation and production
- Reusability management
- Measurement
  - How to make
- Risk management

The Essence of Practice

- Polya suggests:
    1. *Understand the problem* (communication and analysis).
    2. *Plan a solution* (modeling and software design).
    3. *Carry out the plan* (code generation).
    4. *Examine the result for accuracy* (testing and quality assurance).

Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

Object-Oriented Analysis and Design

- Object-oriented analysis
  - emphasizes the finding and describing the objects in the problem domain.
- Object-oriented design
  - emphasizes the definition of software objects and how they collaborate to fulfill the requirements.
- Object-oriented programming
  - the design objects are implemented.

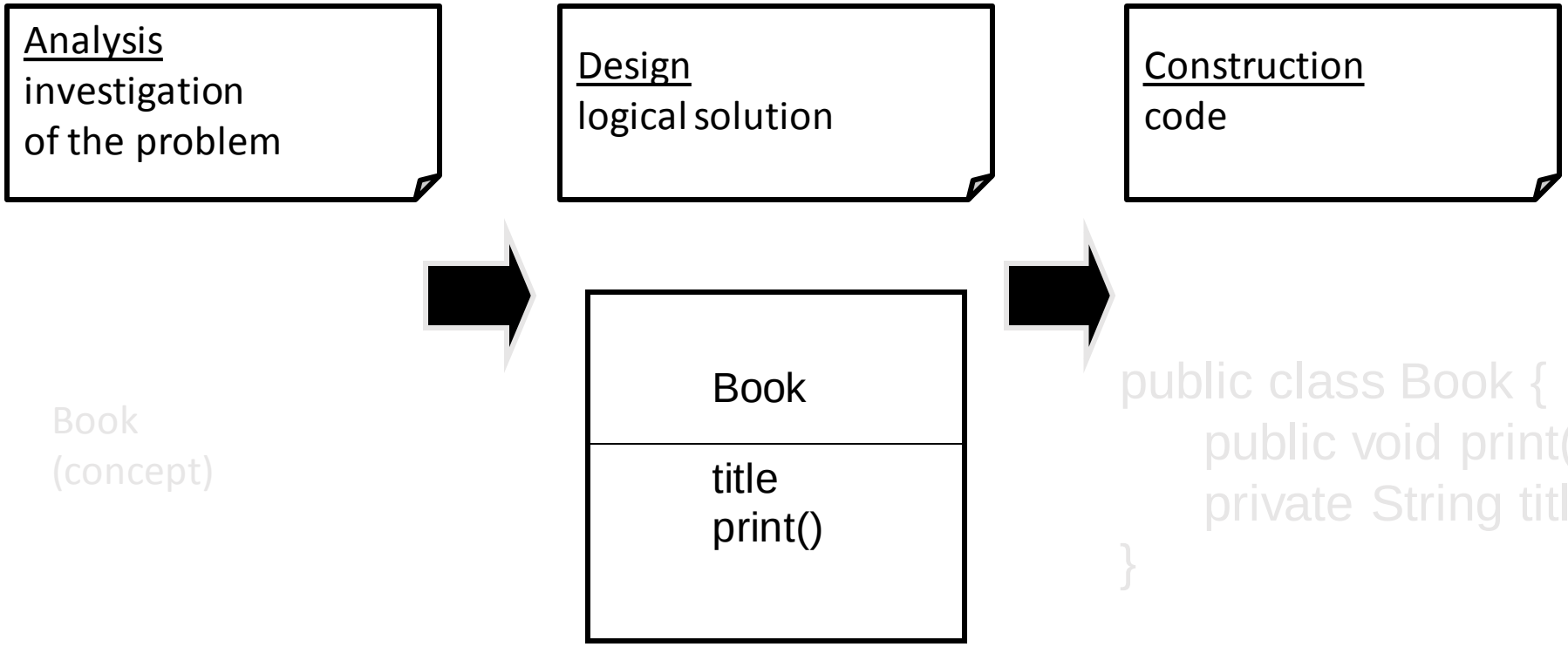
Object-Oriented Analysis

- An investigation of the problem (rather than how a solution is defined)
- During OO analysis, there is an emphasis on finding and describing the objects (or concepts) in the problem domain.
  - For example, concepts in a Library Information System include *Book*, and *Library*.

Object-Oriented Design

- Emphasizes a conceptual solution that fulfils he requirements.
- Need to define software objects and how they collaborate to fulfil the requirements.
  - For example, in the Library Information System, a *Book* software object may have a *title* attribute and a *getChapter* method.
- Designs are implemented in a programming language.
  - In the example, we will have a *Book* class in Java.

From Design to Implementation?



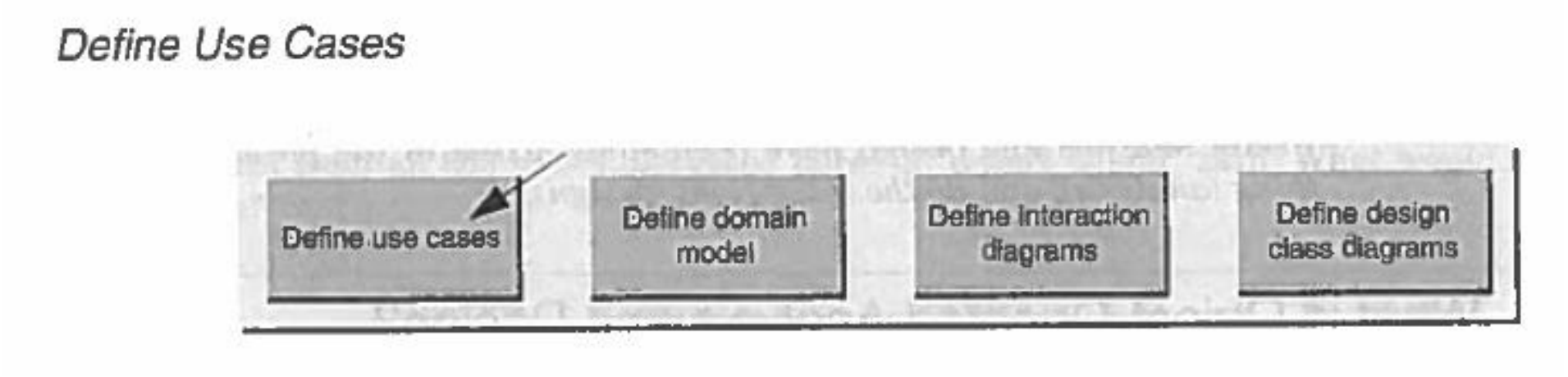
Analysis and Design

- Analysis emphasizes an investigation of the problem and its requirements, not the solution.
  - In this context, it is best described as object analysis, an investigation of the domain objects

- Design emphasizes a conceptual solution that fulfills the requirements, not the implementation.
  - In this context, it is best described as object design.

Use Cases

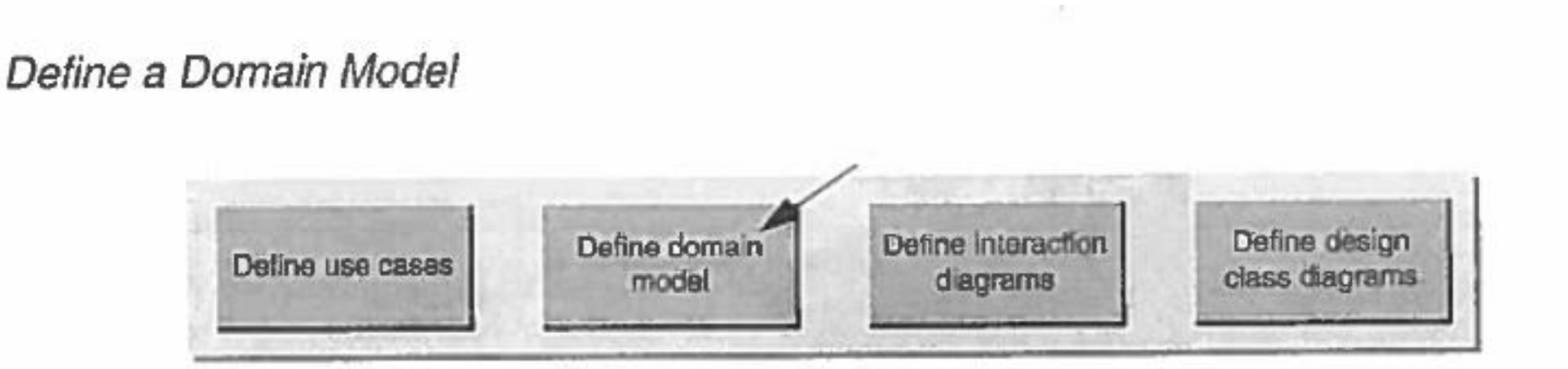
- Use cases are not object oriented artifact
- Very popular and extraordinary useful in describing the requirements and a popular tool in requirements analysis.
- An important part of the unified process (UP)



An Example: The Dice Game

- A Use Case:

  A player picks up and rolls the dice. If the dice face value total seven, the player wins; otherwise, the player loses.



Domain Model

- **a decomposition of the domain model involves an identification of its**
  - **Concepts (conceptual classes)**
  - **attributes**
  - **associations**
- **This decomposition is shown in a domain model**
  - **not software objects**
  - **a visualization of the real-world domain.**

Domain Model

For example, a partial domain model is shown in Figure 1.3.

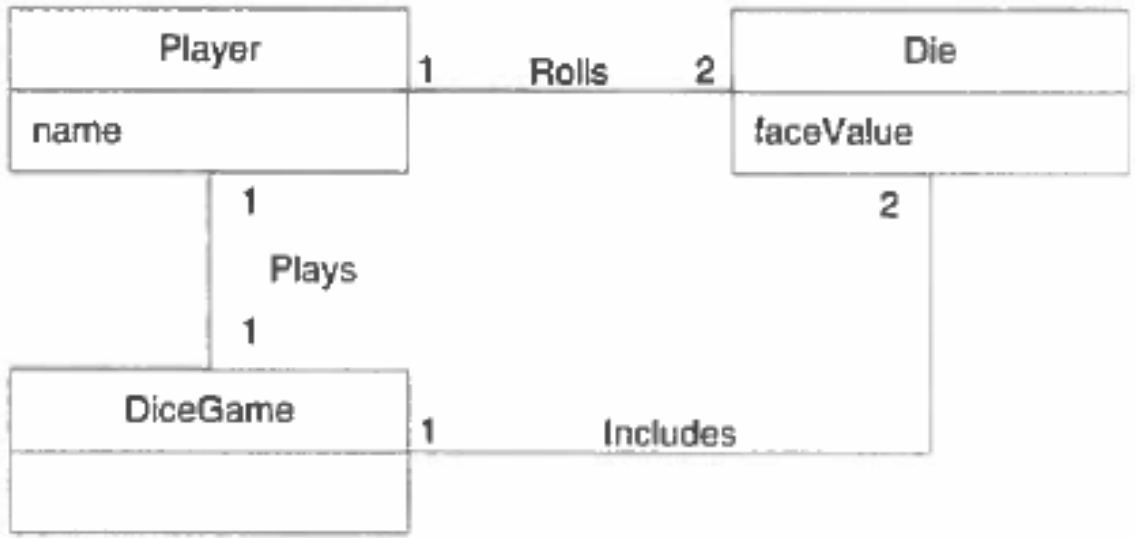
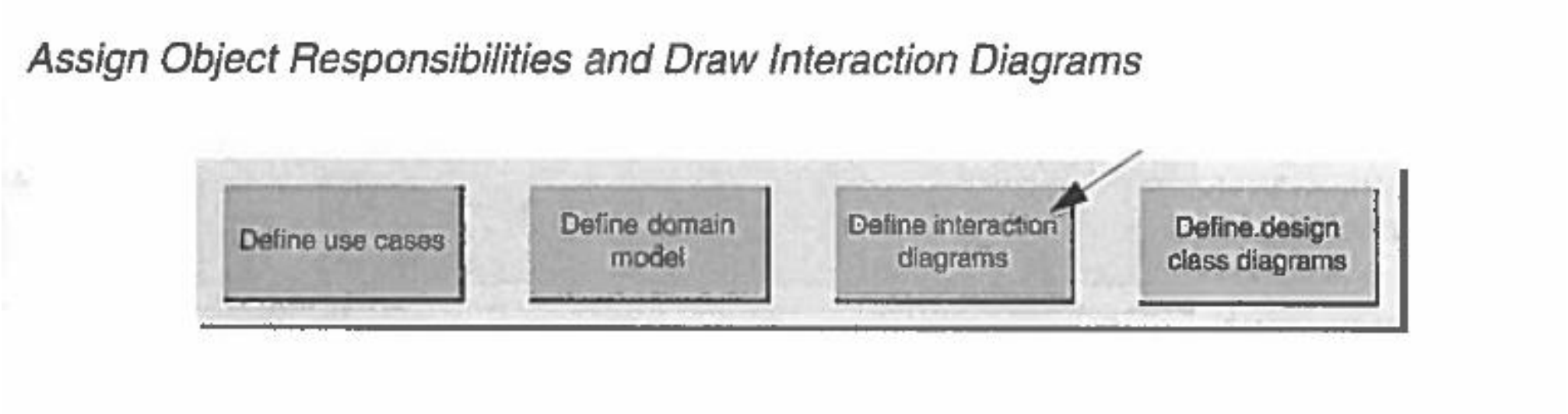


Figure 1.3 Partial domain model of the dice game.

Interaction Diagrams

- objects interact and collaborate with one another.
- Interaction diagrams capture this interaction and show the flow of "messages" from object to object.



Interaction Diagrams

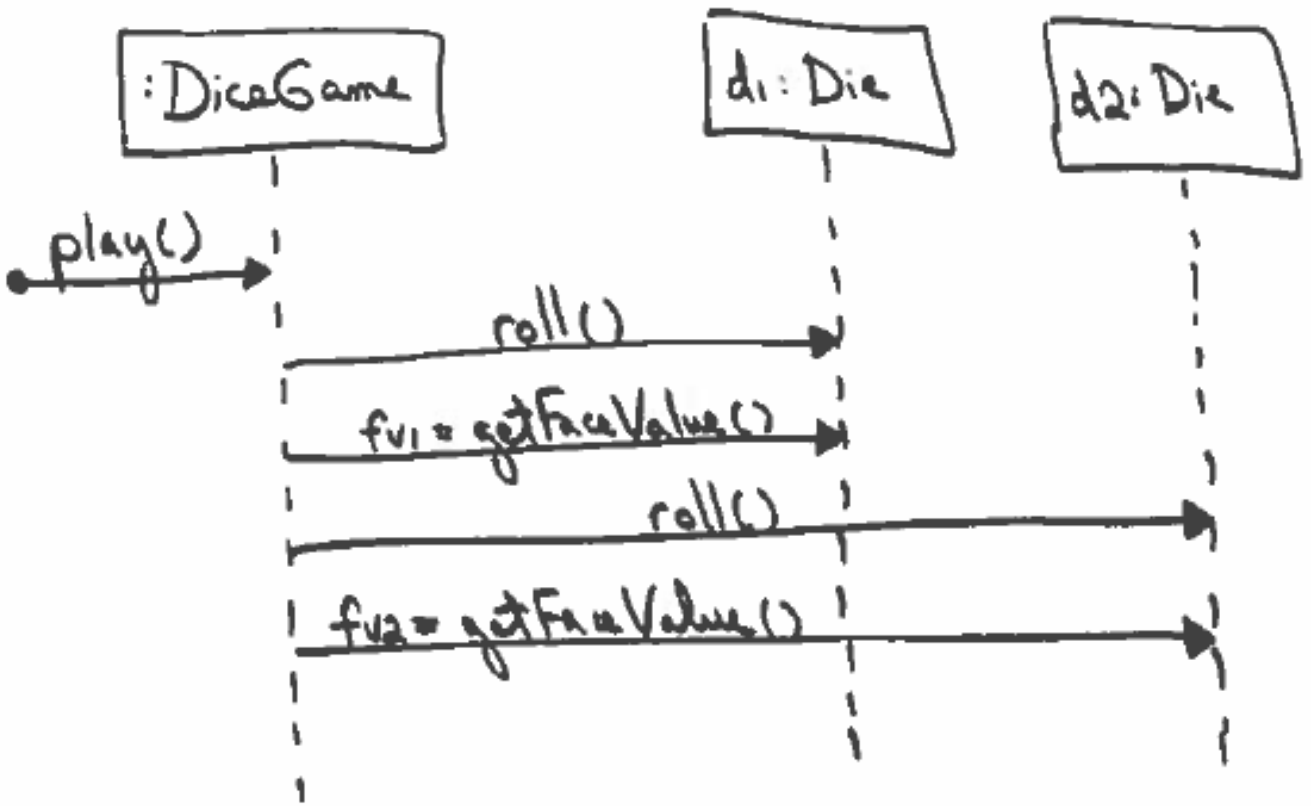
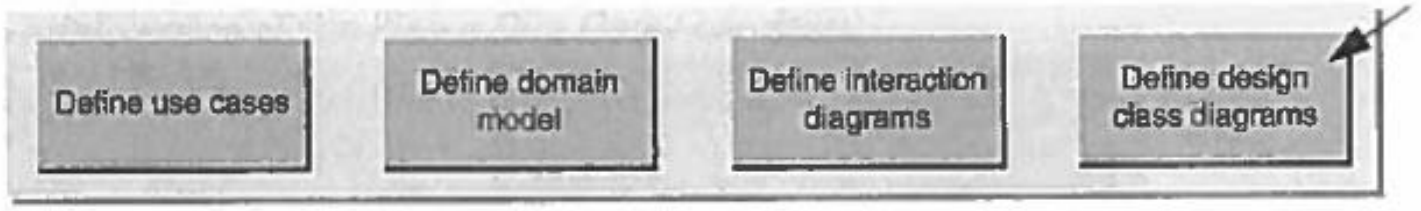


Figure 1.4 Sequence diagram illustrating messages between software objects.

Design Class Diagrams

- Interaction diagrams capture the dynamic view
- Design class diagrams capture the static relationship between classes.
- It illustrates the attributes and methods of a class and the classes relationship to other classes

Define Design Class Diagrams



Design Class Diagrams

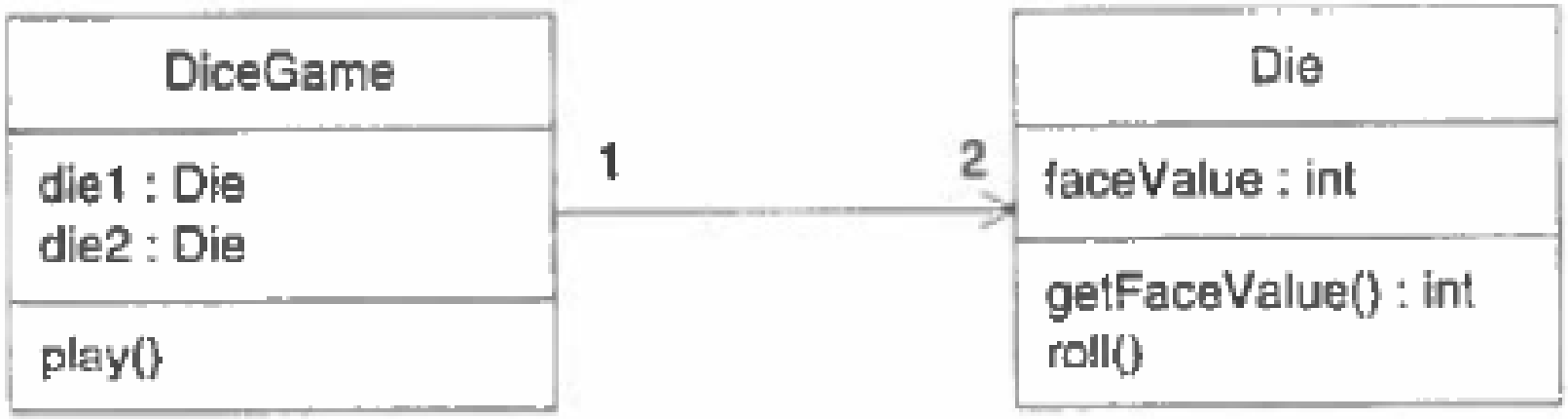


Figure 1.5 Partial design class diagram.

Domain model and Design Class Diagrams

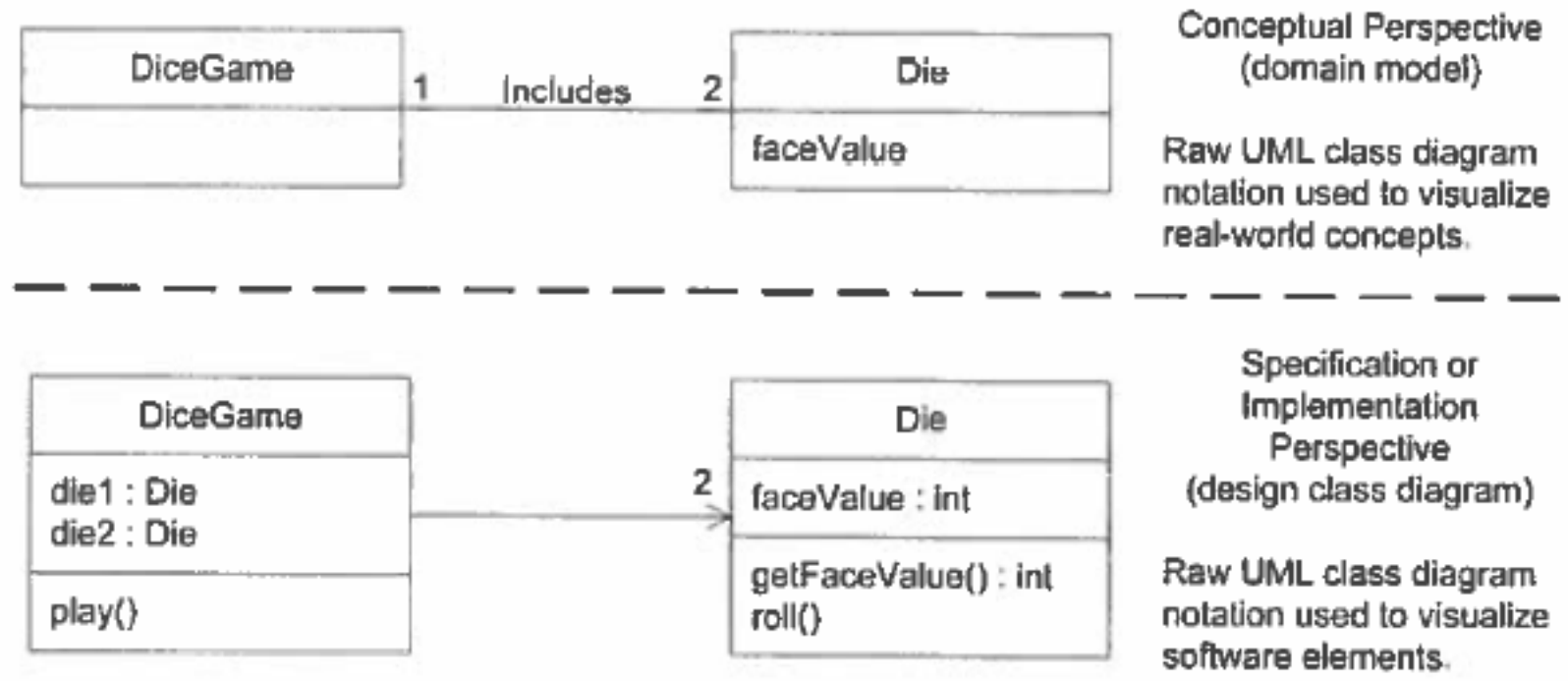


Figure 1.6 Different perspectives with UML.

UML, Patterns

- The Unified Modeling Language is a visual language for specifying, constructing, and documenting the artifacts of systems.
- UML is de facto standard diagramming notation for drawing
  - You can say, that UML is a modeling language for creating diagrams.
  - UML is a standard diagrammatic notation.
  - UML is language-independent
  - UML is not OOA/D or a method
- Patterns – well known, best-practice solutions to common problems