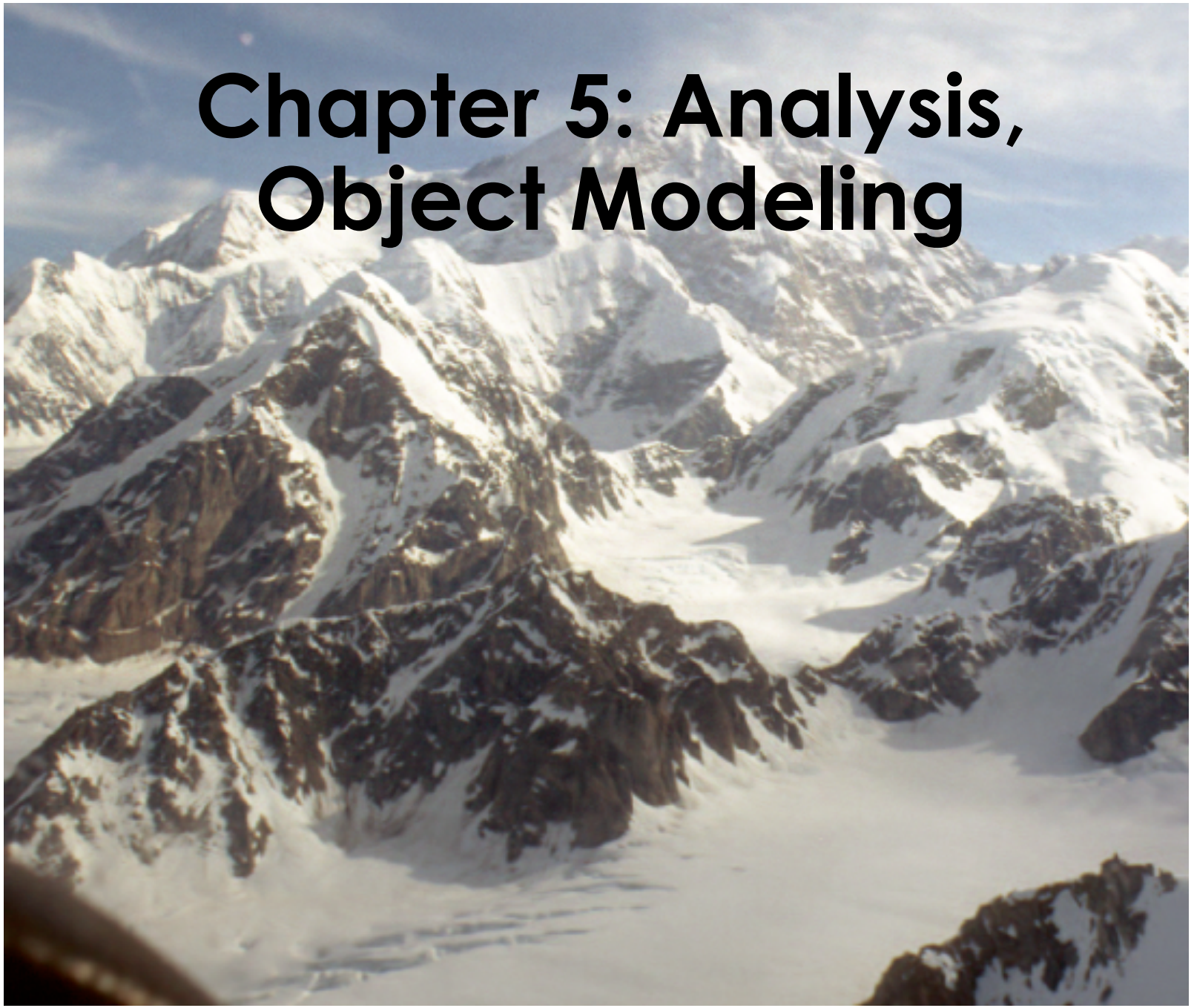
Object-Oriented Software Engineering
Using UML, Patterns, and Java

Chapter 5: Analysis, Object Modeling



Outline

Recall: System modeling = Functional modeling + Object modeling + Dynamic modeling

✓ We looked at: Functional modeling

- Today: Object modeling
  - Activities during object modeling
  - Object identification
  - Object types
    - Entity, boundary and control objects
  - Abott's technique
    - Helps in object identification.

Activities during Object Modeling

Main goal: Find the important abstractions

- Steps during object modeling
  1. Class identification
     - Based on the fundamental assumption that we can find abstractions
  2. Find the associations between classes
  3. Find the attributes
  4. Find the methods

Class Identification

Class identification is crucial to object-oriented modeling

- Helps to identify the important entities of a system

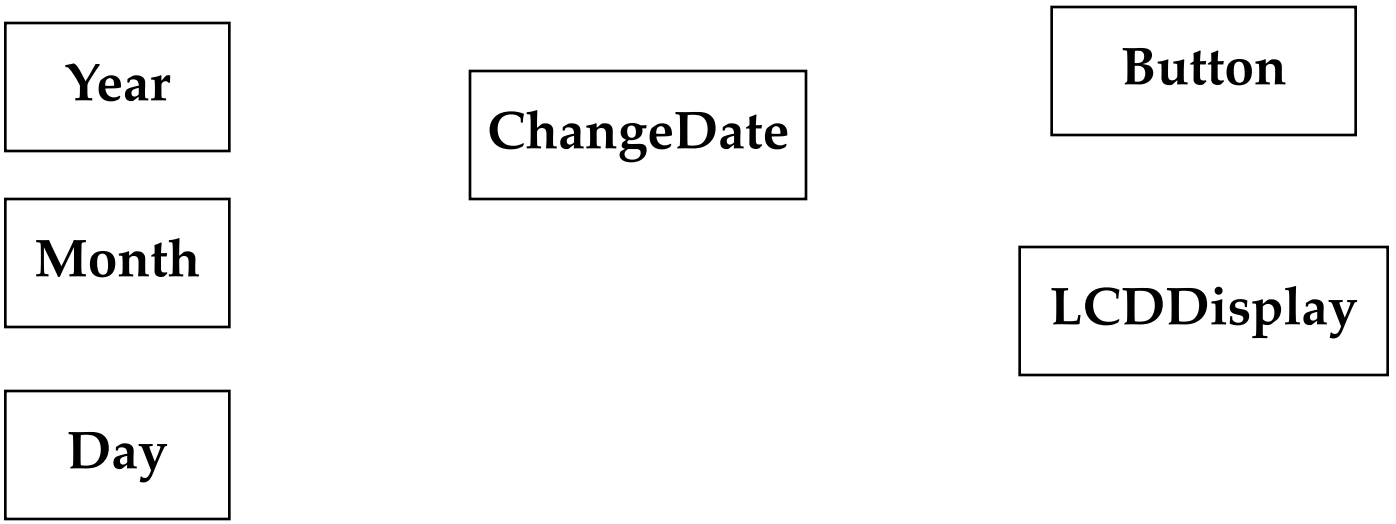
Approaches to Class Identification

- Application domain approach
  - Ask application domain experts to identify relevant abstractions
- Syntactic approach
  - Start with use cases
  - Analyze the text to identify the objects
  - Extract participating objects from flow of events
- Design patterns approach
  - Use reusable design patterns
- Component-based approach
  - Identify existing solution classes.

There are different types of Objects

- Entity Objects
  - Represent the persistent information tracked by the system (Application domain objects, also called "Business objects")
- Boundary Objects
  - Represent the interaction between the user and the system
- Control Objects
  - Represent the control tasks performed by the system.

Example: 2BWatch Modeling



Entity Objects Control Object Boundary Objects

Object Types allow us to deal with Change

- Having three types of object leads to models that are more resilient to change
  - The interface of a system changes more likely than the control
  - The way the system is controlled changes more likely than entities in the application domain

Finding Participating Objects in Use Cases

- Pick a use case and look at flow of events
- Do a textual analysis (noun-verb analysis)
  - Nouns are candidates for objects/classes
  - Verbs are candidates for operations
  - This is also called Abbott's Technique
- After objects/classes are found, identify their types
  - Identify real world entities that the system needs to keep track of (FieldOfficer -> Entity Object)
  - Identify real world procedures that the system needs to keep track of (EmergencyPlan -> Control Object)
  - Identify interface artifacts (PoliceStation -> Boundary Object).

Pieces of an Object Model

- Classes and their instances ("objects")
- Associations between classes and objects
- Attributes
- Operations

Associations

- Types of Associations
  - Canonical associations
    - Part-of Hierarchy (Aggregation)
    - Kind-of Hierarchy (Inheritance)
  - Generic associations

Attributes

- Detection of attributes is application specific
- Attributes in one system can be classes in another system
- Turning attributes to classes and vice versa

Operations

- Source of operations
  - Use cases in the functional model
  - General world knowledge
  - Generic operations: Get/Set
  - Design Patterns
  - Application domain specific operations
  - Actions and activities in the dynamic model

Who uses Class Diagrams?

- Purpose of class diagrams
  - The description of the static properties of a system
- The main users of class diagrams:
  - The application domain expert
    - uses class diagrams to model the application domain (including taxonomies)
      - during requirements elicitation and analysis
  - The developer
    - uses class diagrams during the development of a system
      - during analysis, system design, object design and implementation.

Summary

- System modeling
  - Functional modeling+object modeling+dynamic modeling
- Functional modeling
  - From scenarios to use cases to objects
- Object modeling is the central activity
  - Class identification is a major activity of object modeling
  - Easy syntactic rules to find classes and objects
  - Abbot's Technique
- Class diagrams are the "center of the universe" for the object-oriented developer
  - The end user focuses more on the functional model and the usability.

Dynamic Modeling

- Definition of a dynamic model:
  - Describes the components of the system that have interesting dynamic behavior
- The dynamic model is described with
  - State diagrams: One state diagram for each class with interesting dynamic behavior
  - Sequence diagrams: For the interaction between classes
- Purpose:
  - Detect and supply operations for the object model.
- We also use dynamic modeling for the design of user interfaces

UML Interaction Diagrams

- Two types of interaction diagrams:
  - Sequence Diagram:
    - Describes the dynamic behavior of several objects over time
    - Good for real-time specifications
  - Collaboration Diagram:
    - Shows the temporal relationship among objects
    - Position of objects is based on the position of the classes in the UML class diagram.
    - Does not show time,

How do we detect Operations?

- We look for objects, who are interacting and extract their "protocol"
- We look for objects, who have interesting behavior on their own
- Good starting point: Flow of events in a use case description
- From the flow of events we proceed to the sequence diagram to find the participating objects.

Sequence Diagram

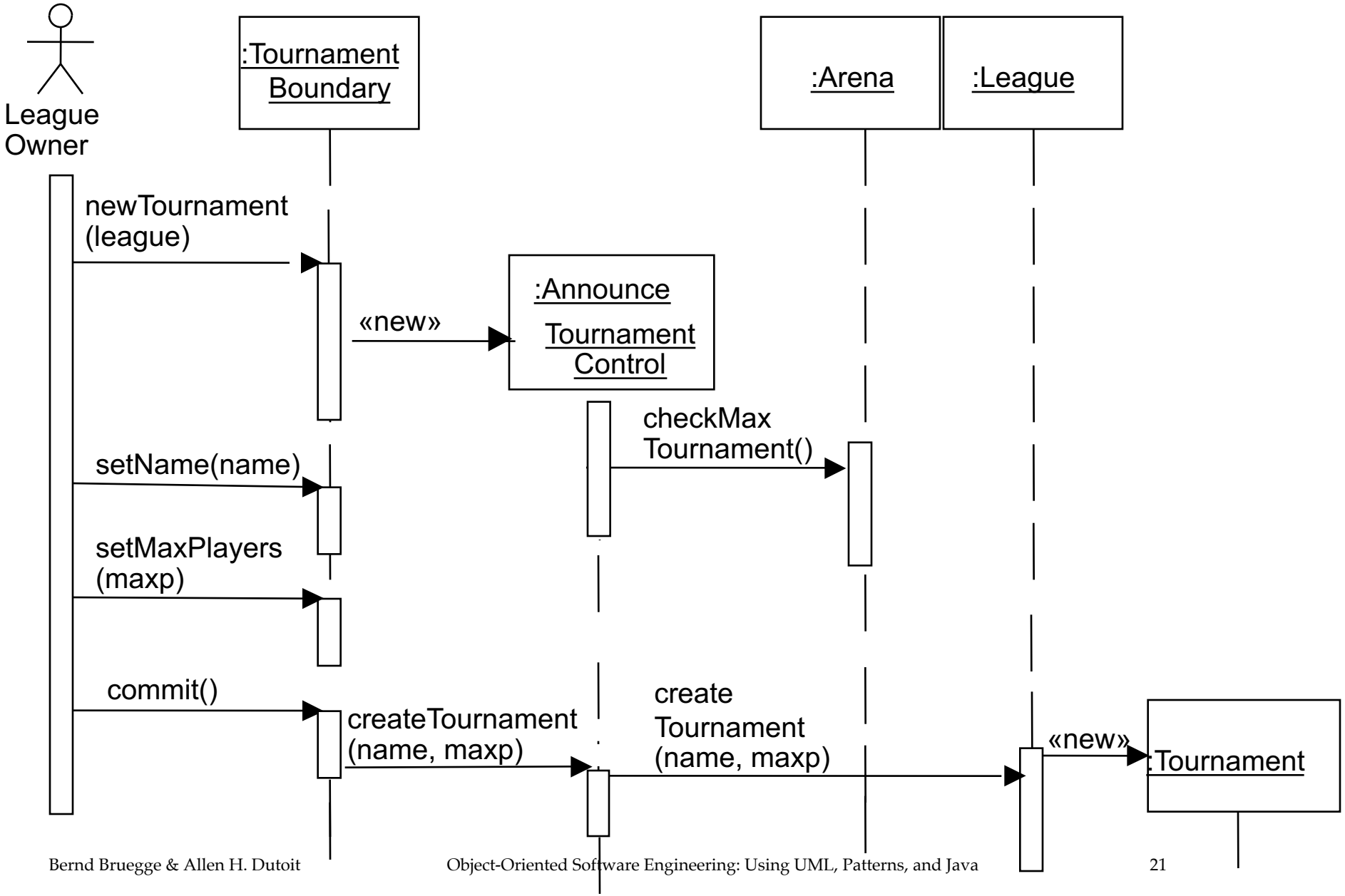
- A sequence diagram is a graphical description of the objects participating in a use case
- Heuristic for finding participating objects:
  - An event always has a sender and a receiver
  - Find them for each event => These are the objects participating in the use case.

Heuristics for Sequence Diagrams

- Layout:
  1st column: Should be the actor of the use case
  2nd column: Should be a boundary object
  3rd column: Should be the control object that manages the rest of the use case

- Creation of objects:
  - Create control objects at beginning of event flow
  - The control objects create the boundary objects
- Access of objects:
  - Entity objects can be accessed by control and boundary objects
  - Entity objects should not access boundary or control objects.

ARENA Sequence Diagram: Create Tournament



Impact on ARENA's Object Model

- Let's assume ARENA's object model contains - at this modeling stage - the objects
  - League Owner, Arena, League, Tournament, Match and Player

- The Sequence Diagram identifies 2 new Classes
  - Tournament Boundary, Announce_Tournament_Control

Model Validation and Verification

- Verification is an equivalence check between the transformation of two models
- Validation is the comparison of the model with reality
  - Validation is a critical step in the development. Process Requirements should be validated with the client and the user.
  - Techniques: Formal and informal reviews (Meetings, requirements review)
- Requirements validation involves several checks
  - Correctness, Completeness, Ambiguity, Realistism

Checklist for a Requirements Review

- Is the model correct?
  - A model is correct if it represents the client's view of the the system
- Is the model complete?
  - Every scenario is described
- Is the model consistent?
  - The model does not have components that contradict each other
- Is the model unambiguous?
  - The model describes one system, not many
- Is the model realistic?
  - The model can be implemented