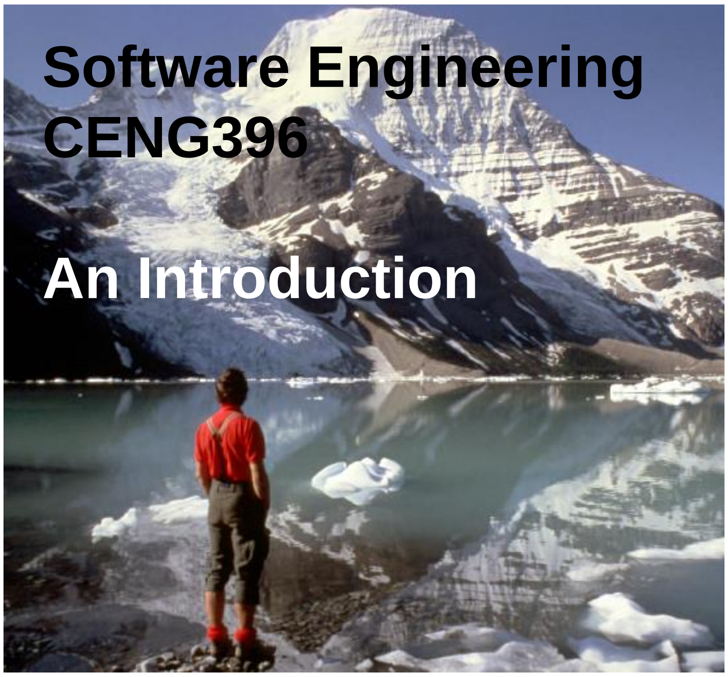
Object-Oriented Software Engineering
Using UML, Patterns, and Java



Software Engineering CENG396

An Introduction

Dealing with Complexity

- Abstraction
- Decomposition
- Hierarchy

1. Abstraction

- Inherent human limitation to deal with complexity
  - The 7 +- 2 phenomena
- Chunking: Group collection of objects
- Ignore unessential details => Models

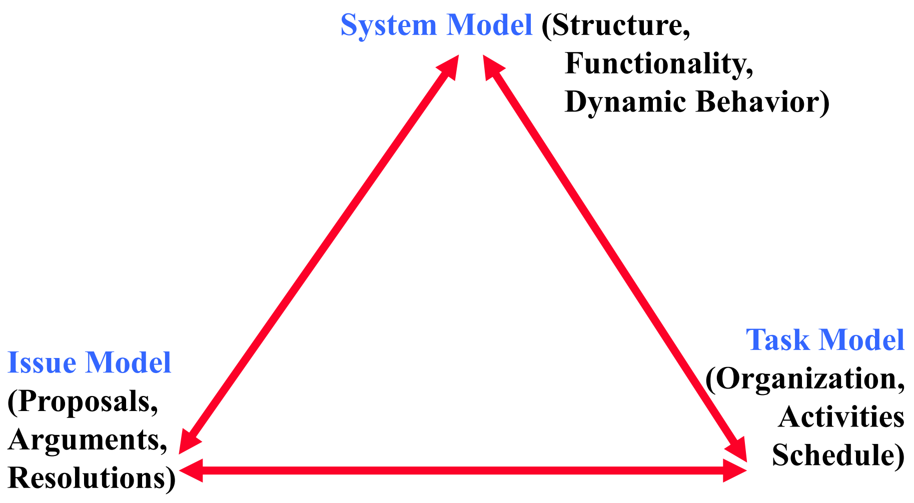
3

Models are used to provide abstractions

- **System Model**
  - **Object Model**: What is the structure of the system? What are the objects and how are they related?
  - **Functional model**: What are the functions of the system? How is data flowing through the system?
  - **Dynamic model**: How does the system react to external events? How is the event flow in the system ?
- **Task Model**
  - **PERT Chart**: What are the dependencies between the tasks?
  - **Schedule**: How can this be done within the time limit?
  - **Org Chart**: What are the roles in the project or organization?
- **Issues Model**
  - What are the open and closed issues? What constraints were posed by the client? What resolutions were made?

4

Interdependencies of the Models

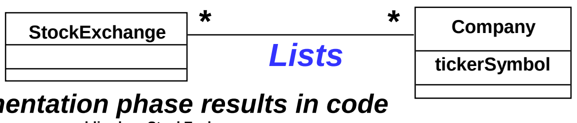

System Model (Structure, Functionality, Dynamic Behavior)
Issue Model (Proposals, Arguments, Resolutions)
Task Model (Organization, Activities Schedule)

5

Model-based Software Engineering: Code is a derivation of object model

Problem Statement: A stock exchange lists many companies.
Each company is identified by a ticker symbol

Analysis phase results in object model (UML Class Diagram):



Implementation phase results in code

```
public class StockExchange
{
   public Vector m_Company = new Vector();
};

public class Company
{
   public int m_tickerSymbol
   public Vector m_StockExchange = new Vector();
};
```

A good software engineer writes as little code as possible

7

2. Decomposition

- A technique used to master complexity (“divide and conquer”)
- Functional decomposition
  - The system is decomposed into modules
  - Each module is a major processing step (function) in the application domain
  - Modules can be decomposed into smaller modules
- Object-oriented decomposition
  - The system is decomposed into classes (“objects”)
  - Each class is a major abstraction in the application domain
  - Classes can be decomposed into smaller classes

Which decomposition is the right one?

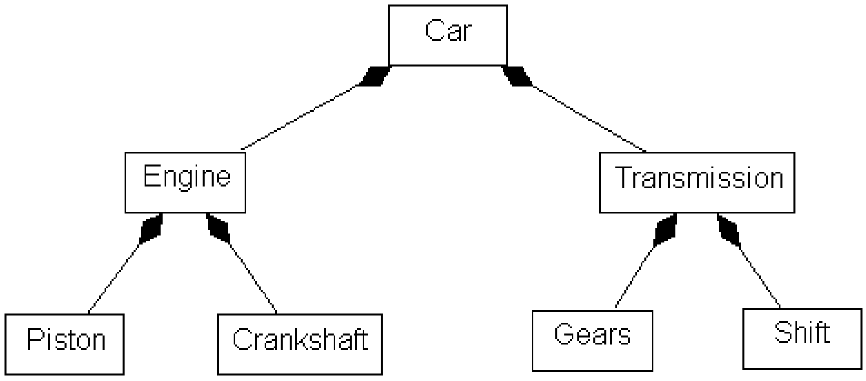
8

3. Hierarchy

- We got abstractions and decomposition
  - This leads us to chunks (classes, objects) which we view with object model
- Another way to deal with complexity is to provide simple relationships between the chunks
- One of the most important relationships is hierarchy
- 2 important hierarchies
  - "Part of" hierarchy
  - "Is-kind-of" hierarchy

9

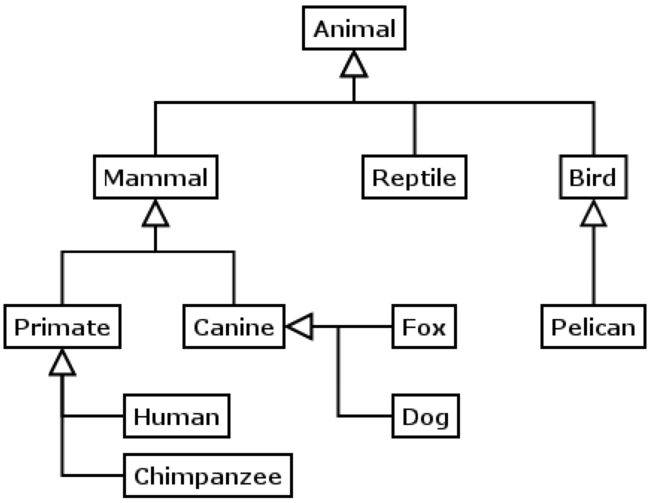
Part of Hierarchy



http://www.conradbock.org/relation4.html

10

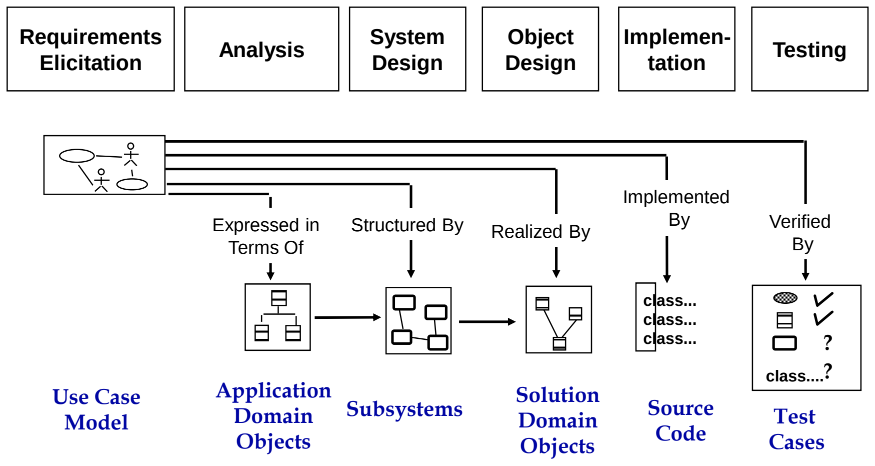
Is-Kind-of Hierarchy (Taxonomy)



http://cs.lmu.edu/~ray/notes/devel/

11

Software Lifecycle Activities ...and their models



Summary

- Software engineering is a problem solving activity
  - Developing quality software for a complex problem within a limited time while things are changing
- There are many ways to deal with complexity
  - Modeling, decomposition, abstraction, hierarchy
  - Issue models: Show the negotiation aspects
  - System models: Show the technical aspects
  - Task models: Show the project management aspects
  - Use patterns/styles: Reduce complexity even further
- Many ways to deal with change
  - Tailor the software lifecycle to deal with changing project conditions
  - Use a nonlinear software lifecycle to deal with changing requirements or changing technology
  - Provide configuration management to deal with changing entities