CS 2212B

Introduction to Software Engineering

Chapter 1

Key Principles – Software Life Cycle - Process

Copyright Notice

Part 4

Introduction to Software Life Cycle

When I'm working on a problem, I never think about beauty. I think only how to solve the problem. But when I have finished, if the solution is not beautiful, I know it is wrong.

— Freeman Dyson

Learning Objectives in this Part

1. To understand what the software life-cycle is

Software Life Cycle

- Software is a complex engineering product and cannot be produced in a single step.
- The development or enhancement of a software system starts as an idea, which becomes a set of requirements, which are represented as models, and perhaps as a prototype, which are transformed into designs, which can be used by developers to implement their code, which is tested against the system requirements.
- The artifacts created in one step become the input to the next step until the software is delivered to the customer.
- The sequence of steps used by these methods is commonly referred to as the Software Development Lifecycle (SDLC.)

Software Lifecycle Activities

- Software Requirements Specification ? *A Communication Activity*
- Analysis ? *A Planning Activity*
- Design ? *A Modeling Activity*
- Implementation ? *A Construction Activity*
- Integration and Testing ? *A Construction Activity*
- Deployment to Production ? *A Deployment Activity*
- Release to Customer ? *A Deployment Activity*
- Maintenance
- Retirement

Functionality / quality of the system.

→ CODE

Corrective
Perfective
Adative.

Deployment and Release Cycles

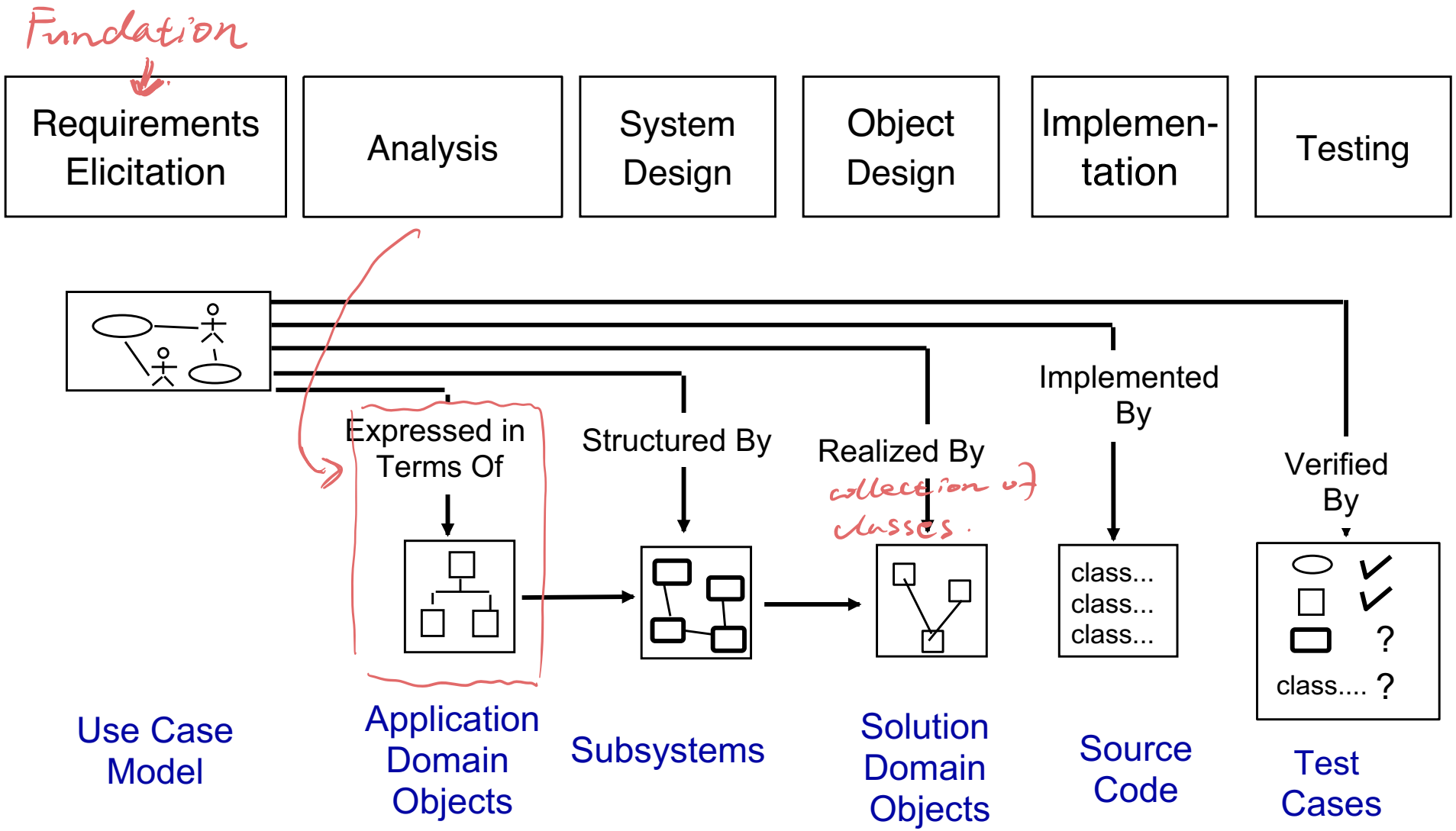
- By the term *Deployment* we refer to the activity whereby the development team *builds* the system (i.e. compiles and integrates system modules) so any functionality built to-date can be tested.

codes that have been made so far.

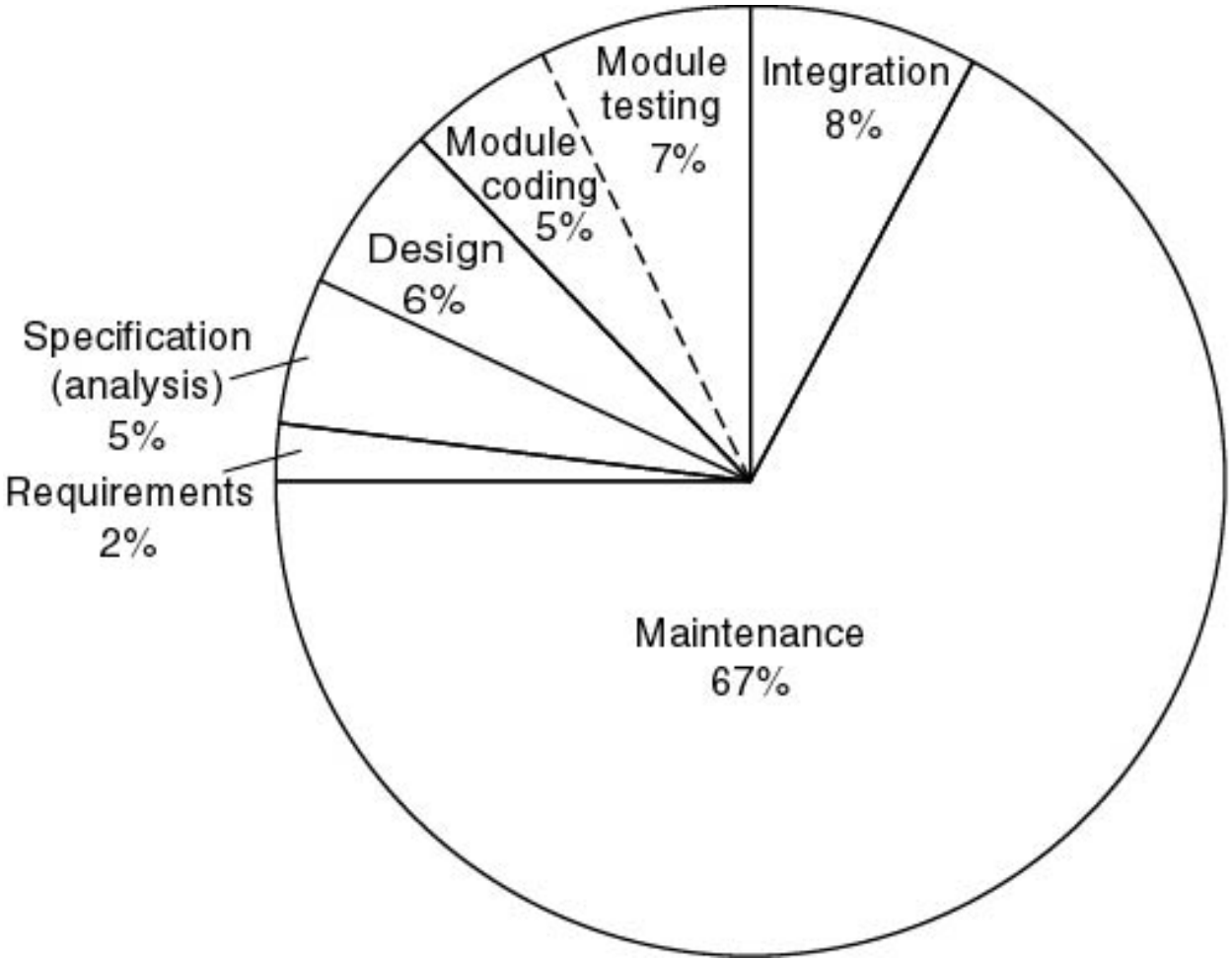
- By the term *Release* we refer to the activity whereby the system is given to its users for use (beta testing or final release).

- Over the past couple of years the develop/integrate/deploy/release cycle is shortened giving rise to what is referred to a *continuous software engineering.*

Software Lifecycle Activities



Indicative average cost distribution for software systems



Part 5

Software Models

The significant problems we face cannot be solved by the same level of thinking that created them. — Albert Einstein

Learning Objectives in this Part

1. To learn about the different types of Models pertaining to Software Engineering

Models

- Each phase in the software life-cycle generates models.
- More specifically:
  - The *Requirements Elicitation* phase generates various models and specifically, ***use case models***, ***sequence diagrams***, **activity diagrams** *and Non-functional specifications*
  - The *Analysis* phase generates ***application domain models*** as well as ***PERT Charts***, ***Gantt Charts*** and ***Organizational Charts***
  - The *Design* Phase generates ***component diagrams***, ***package diagrams***, and ***class diagrams***
  - The *Implementation* phase generates ***source code***
  - The *Testing* phase generates ***test models*** etc.

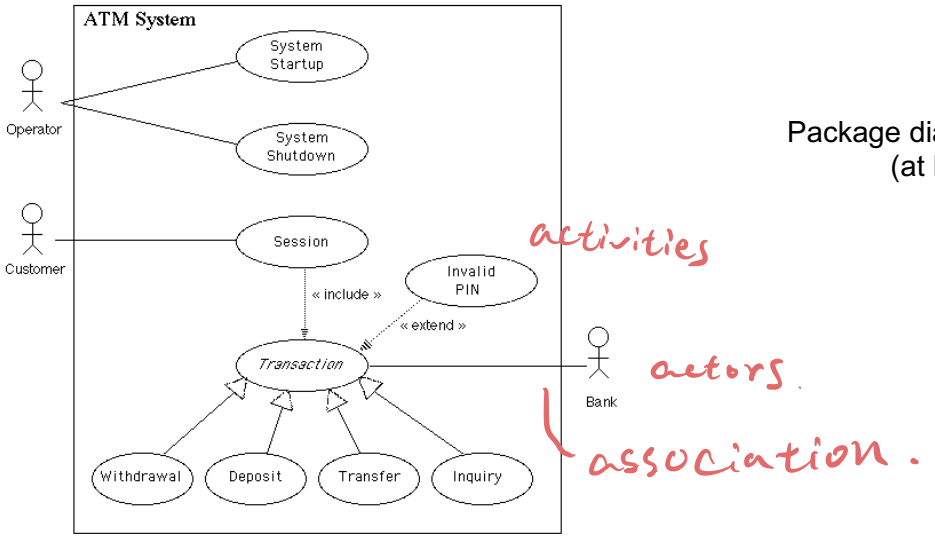
Different Model Categories

- **System Model**:
  - ***Object Model***: What is the structure of the system? What are the objects and how are they related? (see example on next slide)
  - ***Functional Model***: What are the functions of the system? How is data flowing through the system? (see example on next slide)
  - ***Dynamic Model***: How does the system react to external events? How is the event flow in the system ? (see example on next slide)

- **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?

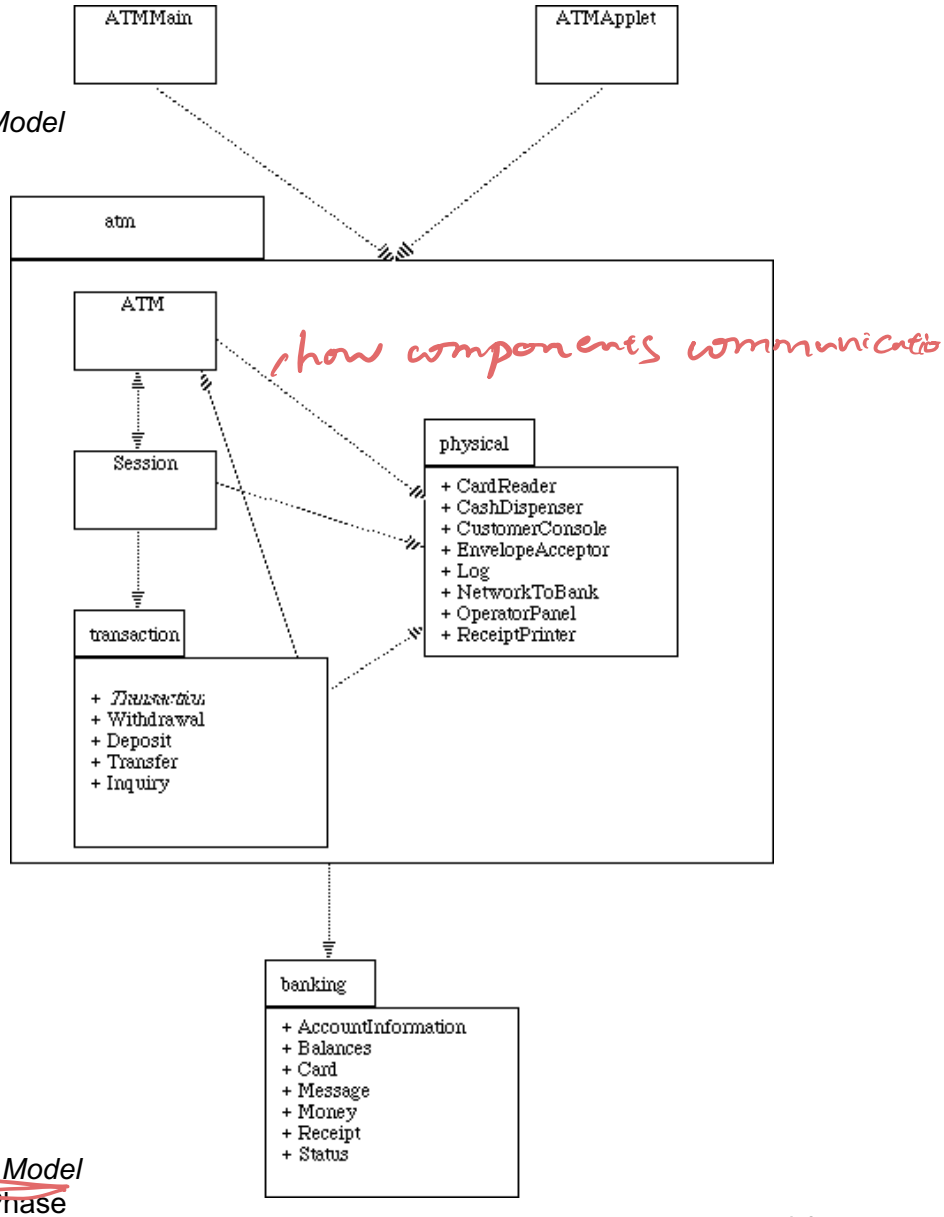
- **Issues Model**:
  - What are the open and closed issues? What constraints were posed by the client? What resolutions were made?

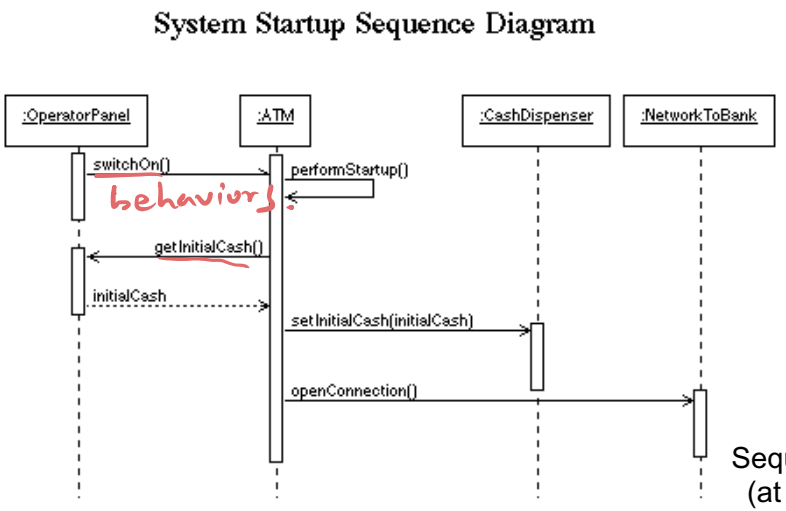
Example Models



Use case model – *Functional Model*
(at Requirements Elicitation Phase)

Package diagram – *Object Model*
(at Design Phase)

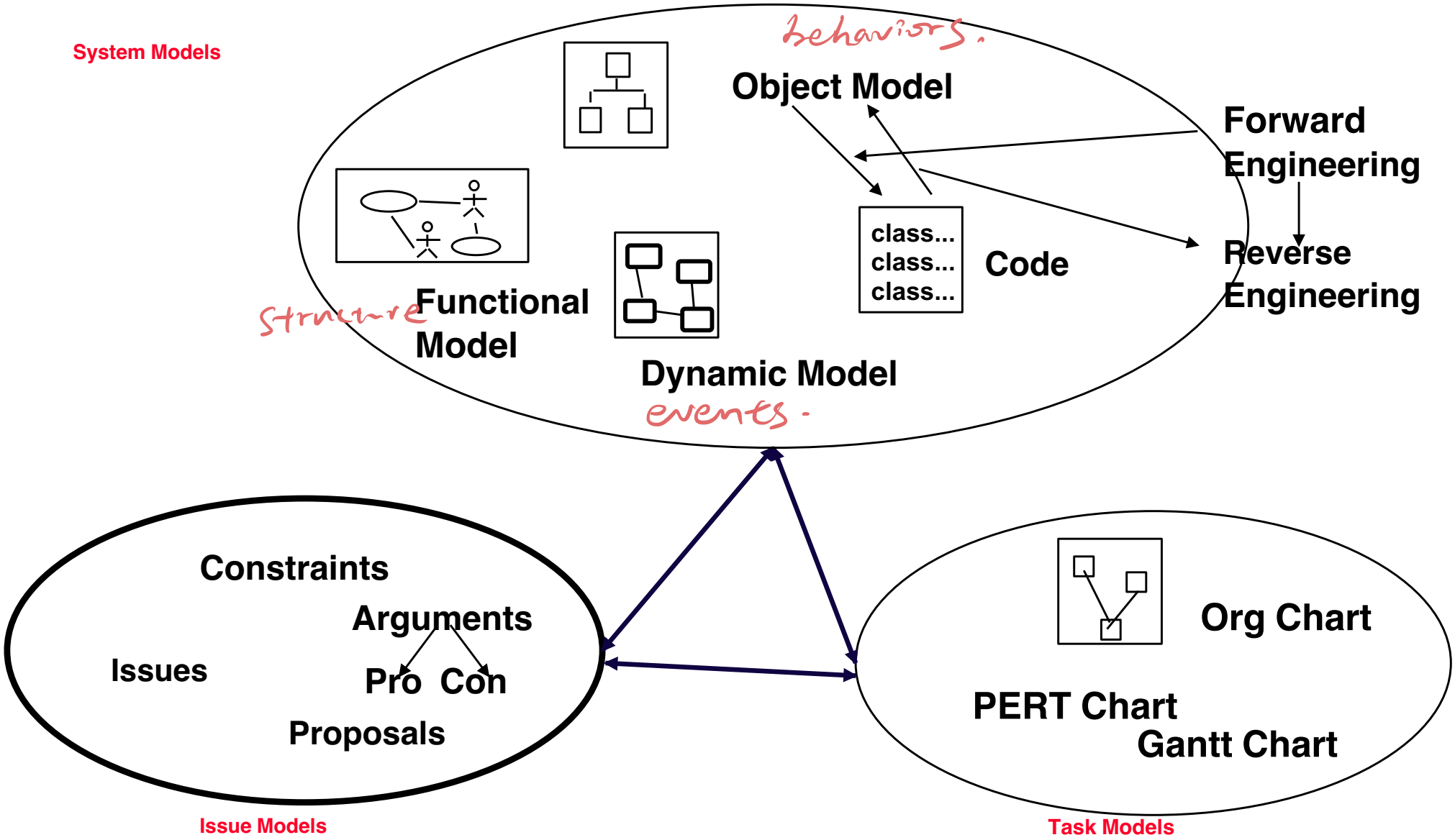




Sequence diagram – *Dynamic Model*
(at Requirements Elicitation Phase)

http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/Intro.html

The Modeling Triangle



Your Turn

- Why do we need models to develop software systems. What are these models used for?

  Making development more organized.

  small/large.
- What is integration phase in the software life cycle?

  Individual develope intergrate large system / System intergrate with application
- Why the maintenance phase is the most costly phase among all the software life-cycle phases?

- Read the content of the following sites:
  - https://en.wikipedia.org/wiki/Systems_development_life_cycle
  - https://ncube.com/blog/software-development-life-cycle-guide
  - https://www.sumologic.com/glossary/software-deployment/