**Unit 1: Basic Software Engineering Concepts (Chapter 1)**

Software Engineering:

* Area of CS which relates to **techniques, methods, practises and tools** for the application of a **systematic, disciplined, quantifiable** approach to the **development, operation and maintenance** of software.
* Helps to develop high quality products within specified budget and time

Method: Settled procedure with a definite and established logical/systematic plan

Methodology: Collection of techniques which are applied throughout phases of project

Technique: Manner and ability in which technical skills are used to achieve result.

Tools: Automated systems that *apply* a technique

Software Lifecycle

1. Software Requirements Specification
2. Analysis
3. Design
4. Implementation
5. Testing and Verification
6. Integration and Deployment
7. Maintenance
8. Retirement

Reusability

* Goal is to design software to be reusable across application domains and designs
* How? Use design patterns and frameworks

Design Pattern: Small set of classes which provide a solution to recurring design problem

Framework: Moderately large set of classes that collaborate to carry out set of responsibilities in an application domain. Example: User interface builder.

Products vs Process

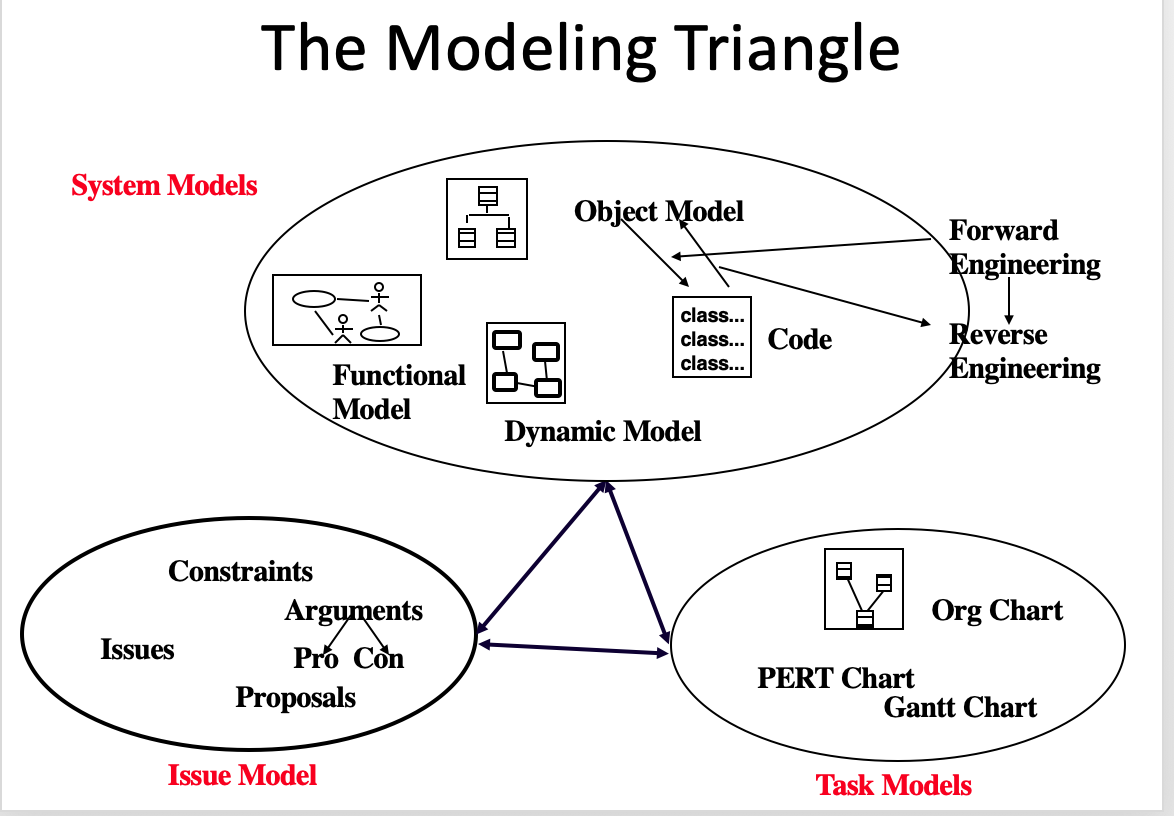
* Products are what bring in revenue
* Process is what are more important:
  + Distinguishes you from your competitor
  + Can bring you to your next product
  + Determines key properties of your product

Challenges of Software Development

1. **Complexity**
   1. New application domains, systems highly integrated, different programming languages
   2. Different frameworks
2. **Quality**
   1. Reliability, Efficiency, Security, Maintainability
3. **Productivity**
   1. New system variants, the gap between demand for new systems, meeting every specific requirement can be difficult
4. **Change**
   1. Systems with business value need to constantly change and be maintained (bugs, changing platforms, adding new functions)
5. **Maintenance and Legacy Systems**
   1. Maintenance: adaptive, perfective, corrective, preventive
   2. Legacy Systems: Long lifetime of software systems, software has to be continuously adapted to new and changing requirements

Dealing with Complexity

1. **Abstraction**
   1. Models provide abstractions
   2. Dealing with different models (object, functional and dynamic) can help you figure out how to expand on the system
   3. Task Models (PERT chart, schedule and Org chart) help understand the roles in the project or tasks
   4. Figure out the issues going on (issue model)



1. **Decomposition**
   1. Technique which breaks down the system into modules to understand the steps in an application
   2. Object-oriented decomposition: System is decomposed into classes which detail the application domain more clearly
2. **Hierarchy**
   1. Abstractions and decomposition lead to chunks depicted in the model
   2. The Hierarchy provides simple relationships between these chunks
   3. Hierarchies allow for inheritance of code and featured from one chunk to another in the hierachy