Software Engineering – Fall 2015
(CSC 4350/6350)
TR. 5:30 pm – 7:15 pm

Rao Casturi
09/03/2015
http://cs.gsu.edu/~ncasturi1
Drop Box Folder Structure

- Includes Individual Bio-Data Information

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Project Contract</td>
<td></td>
</tr>
<tr>
<td>2. Requirement Elicitation</td>
<td>--</td>
</tr>
<tr>
<td>3. System Analysis</td>
<td>--</td>
</tr>
<tr>
<td>4. Object Design</td>
<td>--</td>
</tr>
<tr>
<td>5. Project Rational (All Phases)</td>
<td>--</td>
</tr>
<tr>
<td>6. Testing (Code Included)</td>
<td>--</td>
</tr>
<tr>
<td>7. Final Project</td>
<td>--</td>
</tr>
</tbody>
</table>
Pending Items Due by 09/03/2015

Students:

– Team leader – Contract, Group Names, and Roles
– Project Charter / Project Details
– Project team formulation (4 to 5 members)

Instructor:

– Reply emails for assignments (*In progress*)
What is Software Engineering?

- Software Engineering
- Problem Solving
- Knowledge Acquisition
- Rationale
- Modeling

GSU: Software Engineering - CSC4350/6350 - Rao Casturi
Software engineering concepts depicted as a UML class diagram. [Bruegge, Dutoit]

Projects – Activity – Tasks and Work Product
Work Product:

- Is an artifact produced during the development
- System
- Specification Document
- Operational Manual
- Status Reports
- Testing Manual
Roles and Responsibilities

• **Client**
  – Provide high level requirements
  – Scope
  – Fund
  – Set delivery time, quality

• **User**
  – Provide knowledge about the system
  – Feedback
  – Testing

• **Project Manager**
  – Overall management
  – Face to the Client

• **Developer**
  – Construction of the system
  – Testing

• **Technical Writer**
  – Documentation
  – Reference manual

• **Support**
  – Product support
  – Installation
Functional and Non Functional Requirement

• Functional
  – Specification a **system should** support
  – Need to have

• Non Functional
  – **Constraint** on the System
  – No direct relation to the function of the system
Software Engineering Development Activities

• Requirement Elicitation (Gathering)
• Analysis
• System Design
• Object Design
• Implementation
• Testing
Managing Software Development

• Communication
  – Time consuming activity
  – Critical for the project

• Rational Management
  – Justification – Complex activity

• Software Configuration
  – Version controls
  – Maintenance

• Project Management
  – Art

• Software Life Cycle
  – Putting it all together is the life cycle of the Software Development
Questions
?

GSU: Software Engineering - CSC4350/6350 - Rao Casturi
Introduction to UML

• What is UML?
  – Unified Modeling Language
  – History
  – Why is it so important in Software Engineering?
  – Fundamental Notations of UML
Evolution of UML

1990
Methodologies proliferate
Booch '91
Booch '93
Ada/Booch

State Charts
Harel 1987

1995
Mature practice
"3 amigos"
OOPSLA '95

UM 0.8
UML 0.9

1997
Accepted by OMG Nov. 97
Standardization

UM 1.1
UML 1.3
UML 1.4

March 2003

UML 1.5

2005
Language proliferate

UML 2.0

2007
UML 2.1.2
SysML 1.1
BPMN 1.1

2008
UML 2.2

Reference Wikipedia

GSU: Software Engineering - CSC4350/6350 - Rao Casturi
System Development

Output or produces

Functional Model
- Use Case Diagrams

Object Model
- Class Diagrams

Dynamic Model
- Interaction, State Machine, Activity Diagrams
Basic Notations of UML

- Use Case Diagrams
- Class Diagrams
- Interactive Diagrams
- State Machine Diagrams
- Activity Diagrams
1. Use Case Diagram:

Definitions:
- Use Case:
  - Statement of functionality required by software
  - 3 Main components of Use Case
    - ACTOR, ACTION, SUBJECT
  - Use Case is written in specific format
  - Use Case (UC) is a Functional Requirement

Subject represents the object on which the Action will take place
Example of an Use Case

1. **Portfolio Manager**
   - Balance Portfolio
   - Risk Analysis
   - Set Risk Limits
   - Price The Bond
   - Broker Quote

2. **Risk Manager**
   - Valuation of the Bond

3. **Trader**
   - End of Day Holdings

4. **Data Warehouse**

(System Boundary)

(Use Case)

GSU: Software Engineering - CSC4350/6350 - Rao Casturi
# Use Case Name

**UC_001_Bond_Risk_Analysis**

## Actors/Participants

Risk Manager, Trader

## Flow of Events/Scenario

1. Find the historical P&L of the bond
2. Depending on the P&L data decide if we can add this bond to the portfolio or not.
3. Set Risk Target if you need to add to the portfolio

## Entry Condition

When a Portfolio Manager requests a Bond to be analysed

## Exit Condition

The Risk appetite set

## Quality constraints

The Risk Limit set should be in line with the other similar instruments

---

**Relationships to Reduce Complexity:**

- Communication Relationships (Solid Line)
- <<Include>> (Dotted Line)
- <<Extend>> (Dotted Line)
- Inheritance (Solid Line with Triangle Head)
2. Class Diagram:

- Describes the structure of the system in terms of Classes and Objects
- Classes: - Collection of objects
- Objects: - The Entities to capture the state and behavior of the system
- Consists of 3 Components

<table>
<thead>
<tr>
<th>Class / Object Name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Methods or operations</td>
</tr>
</tbody>
</table>

1. Link: Connection between 2 or more classes
2. Association: Relation between 2 or more classes
   - Roles, Multiplicity (1-1,1-m,m-n)
3. Aggregation: Hierarchical model
4. Inheritance: Relationship to Root Class to many specialized classes
   - Super Class – Sub Class
3. Interaction Diagrams:

- Capture Communication between objects
- Messages between objects (Solid arrows)
- Timing Diagrams or Sequence Diagrams
- X – Axis Objects Y- Axis Time

![Interaction Diagram](image)
4. State Machine Diagrams:
- Describes the sequence of states an object can go through when triggered by an external event.
- Active, Inactive, Closed etc.
Activity Diagrams

5. Activity Diagrams:

- Technique to describe any procedural logic or business process or work flow
- Like flow charts but support parallel behavior
- Forks, Joins, Decisions
- Swim lane diagrams
Questions

?