SS ZG653 (RL 8.1): Software Architecture

Documenting Architecture with UML

Instructor: Prof. Santonu Sarkar

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

September 6, 2022

Unified Modeling Language (Introduction)

- **Modeling Language for specifying, Constructing, Visualizing and documenting software system and its components**
- **Model -> Abstract Representation of the system [Simplified Representation of Reality]**
- **UML supports two types of models:**
  - **Static**
  - **Dynamic**

UML

- Unified Modeling Language is a standardized general purpose modeling language in the field of object oriented software engineering

- The standard is managed, and was created by, the Object Management Group.

- UML includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems

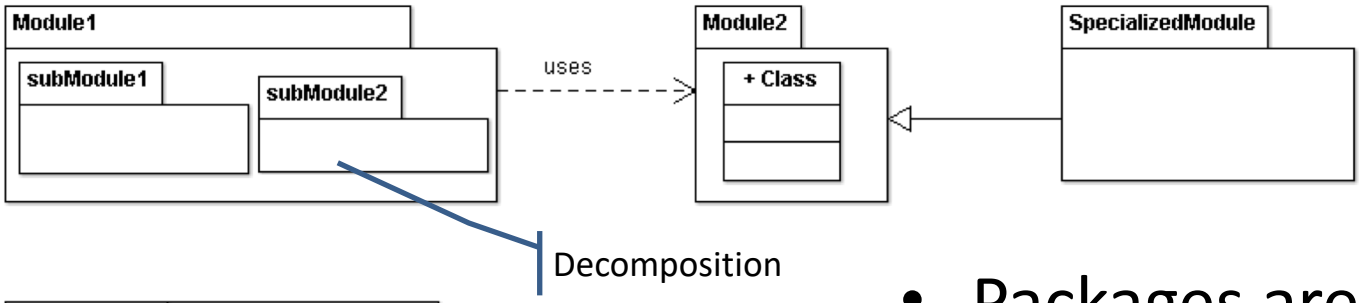
Documenting Architecture

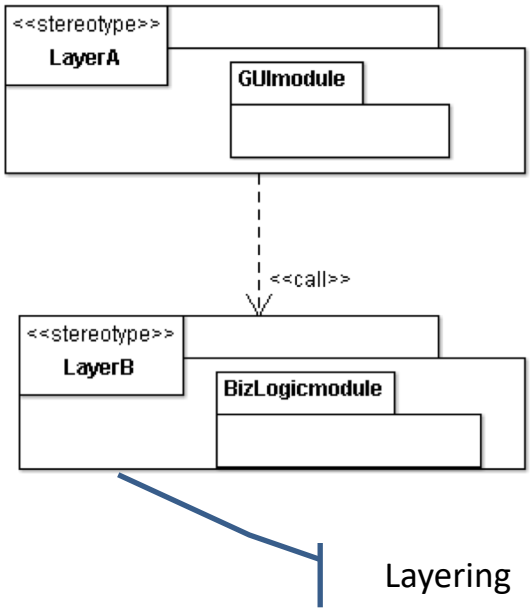
- UML has not been designed specifically for architecture, though practitioners use UML for architecture description
  - It is upto the architect to augment UML for architecture
  - UML provides no direct support for module-structure, component-connector structure or allocation structure

Three Structures- Recap

- Module Structure
  - Code units grouped into modules
  - Decomposition
    - Larger modules decomposed into smaller modules
  - Use
    - One module uses functionality of another module
    - Layered
      - Careful control of uses relation

Illustration- Module Views



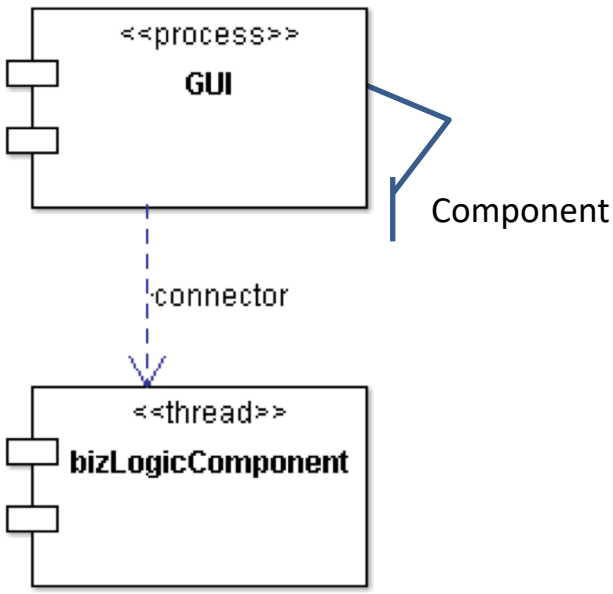


- Packages are used as modules
- Other approaches
  - One can use class, or interface to denote a module
- Relations
  - One can use various other UML relations to denote uses, layering or generalization

Three Structures- Recap

- Component-Connector Structure
  - Processes
    - Components are processes and relations are communications among them
  - Concurrency
    - Relationships between components- control flow dependency, and parallelism
  - Client-Server
    - Components are clients or servers, connectors are protocols
  - Shared data
    - Components have data store, and connectors describe how data is created, stored, retrieved

Illustration- CNC Views

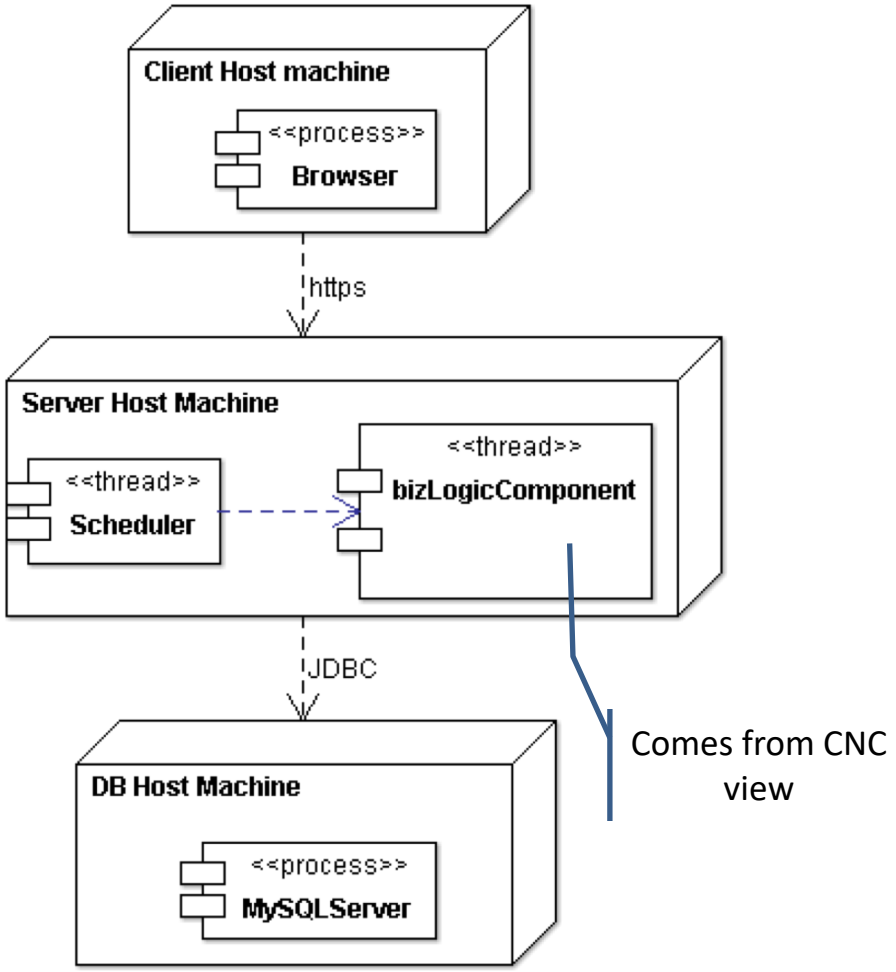


- No standard representation exists
- UML components are used with stereotypes
- Other approaches
  - One can use class, interface, or package to denote a component
- Relations
  - One can use various other UML relations such as association class

Three Structures- Recap

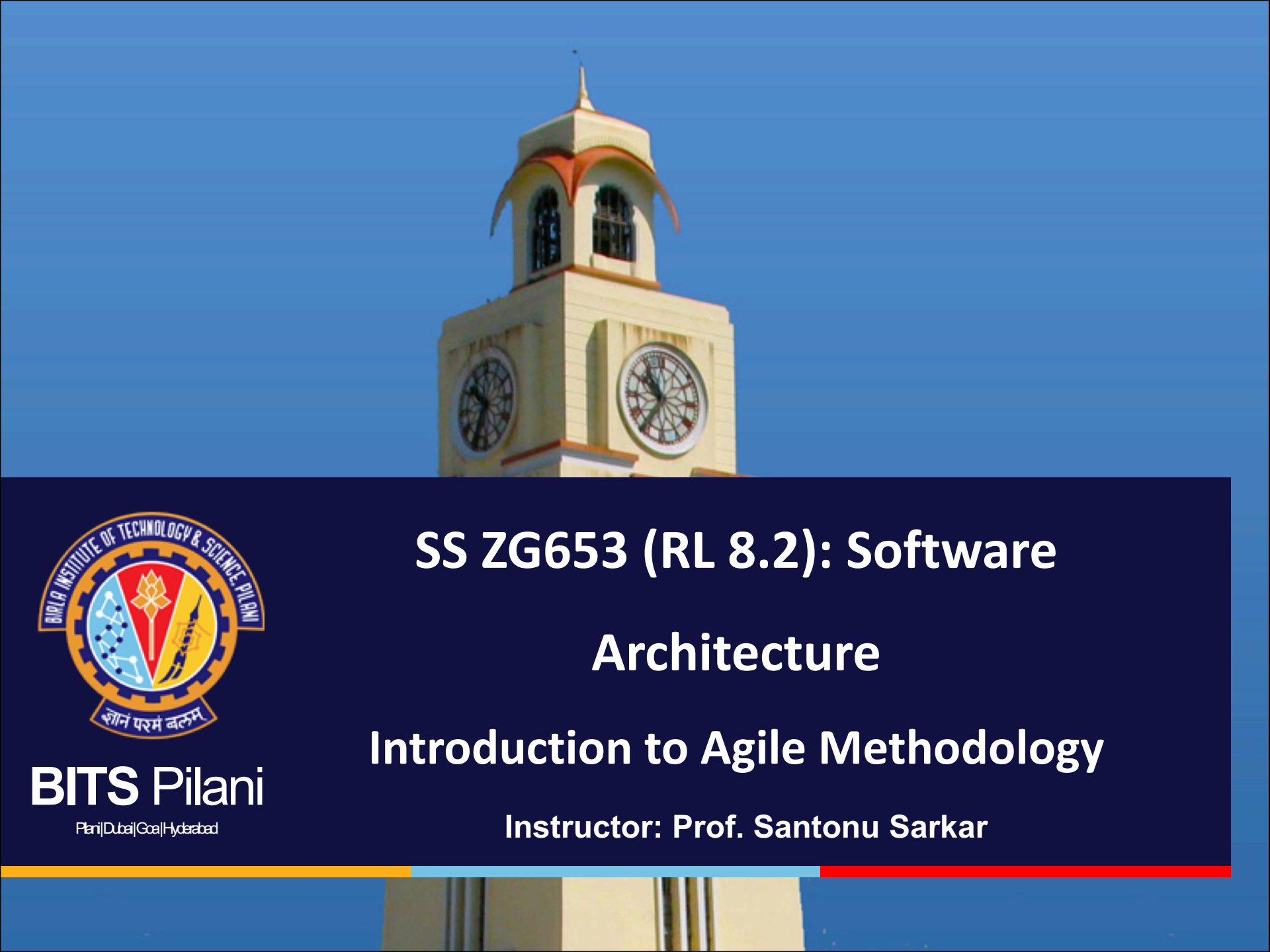
- Allocation Structure
  - Deployment
    - Units are software (processes from component-connector) and hardware processors
    - Relation means how a software is allocated or migrated to a hardware
  - Implementation
    - Units are modules (from module view) and connectors denote how they are mapped to files, folders
  - Work assignment
    - Assigns responsibility for implementing and integrating the modules to people or team

Illustration-Allocation Views



- UML Deployment diagram is a good option for deployment structure
- No specific recommendation for work assignment and implementation

Thank You



SS ZG653 (RL 8.2): Software Architecture

Introduction to Agile Methodology

Instructor: Prof. Santonu Sarkar

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

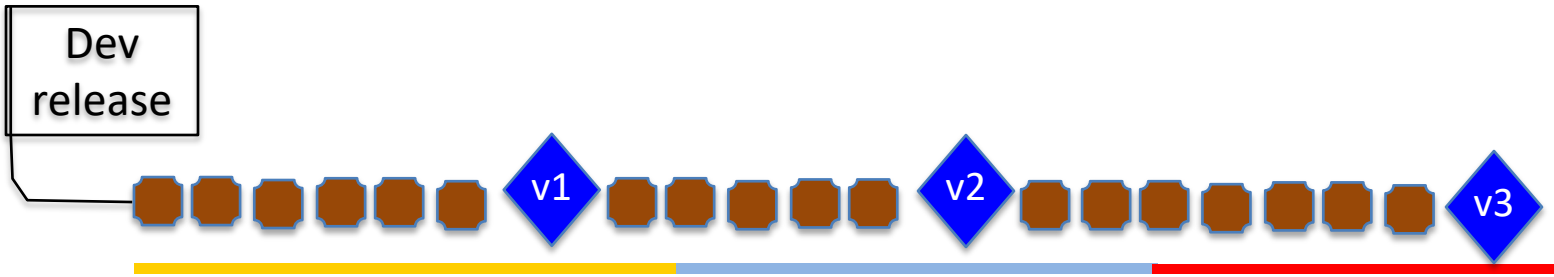
What is Agile Methodology

- Collaborative
  - Forms a pair for any development task to avoid error
  - Involves stakeholders from the beginning
- Interactive and feedback oriented
  - Teams interact frequently
  - Quick, and repeated integration of the product
  - Constant feedback from the stakeholder (customer)
- Iterative
  - Requirement, design, coding, testing goes through many iterations each having short duration
  - Refactoring is a part of the development process
- Test driven
  - Before building the component, define the test cases
  - Continuously test

– Scott Amber, Kent Beck

A Brief Overview

- There are 7 disciplines performed in an iterative manner
- At each iteration the software (or a part of the software) is built, tested
- Software architecture is more “agile” and it is never frozen
- UML based modeling is performed



Discipline Overview

- Model
  - Business Model
  - Analysis and Design (Architecture)
- Implementation
- Test
- Deployment
- Config Management
- Project Management
- Environment

Steps of Architecture Modeling in Agile

- Feature driven
  - Prioritize. Elaborate critical features more
- Model the architecture (UML)
- Suggested viewpoints for Agile
  - Usage scenarios
  - User interface and system interface
  - Network, deployment, hardware
  - Data storage, and transmission
  - Code distribution
- Suggested quality concerns
  - Reuse
  - Reliability, availability, serviceability, performance
  - Security
  - Internationalization, regulation, maintaince

Class Responsibilities and Collaborators (CRC) Card

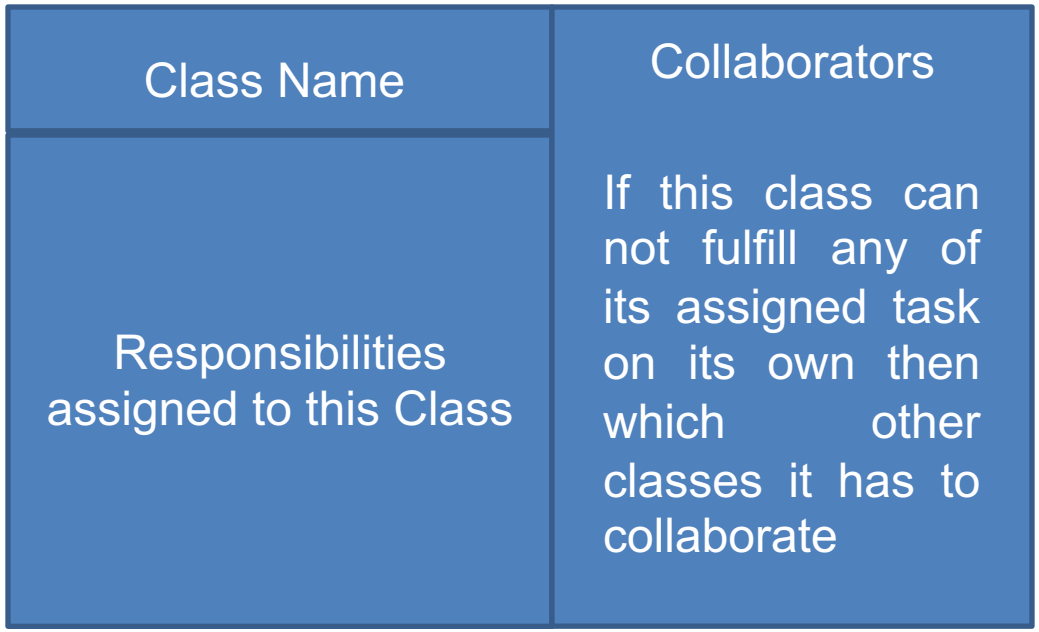
What

- It is a physical (electronic) card
- One card for one class
  - Indicates the responsibilities of a class
- Collaboration
  - Sometimes a class can fulfill all its assigned responsibilities on its own
  - But sometimes, it needs to collaborate with other classes in order to fulfill its own responsibilities

Why?

- A good technique to identify a class and its responsibility during functional architecture design
- Highly collaborative and interactive process for a team of designers
- The team can do it fast

CRC Card


Class Name
Responsibilities assigned to this Class
Collaborators
If this class can not fulfill any of its assigned task on its own then which other classes it has to collaborate

CRC Card Example 1

```
class Box
{
private double length;
private double width;
private double height;
Box(double l, double w, double h)     {     length = l; width = w; height = h;     }
public double getLength()             {     return length;                         }
public double getWidth()              {     return width;                          }
public double getHeight()             {     return height;                         }
public double area()                  {     return 2*(length*width + width * height + height * length);     }
public double volume()                {     return length * width * height;                                  }
} // End of class BOX
```

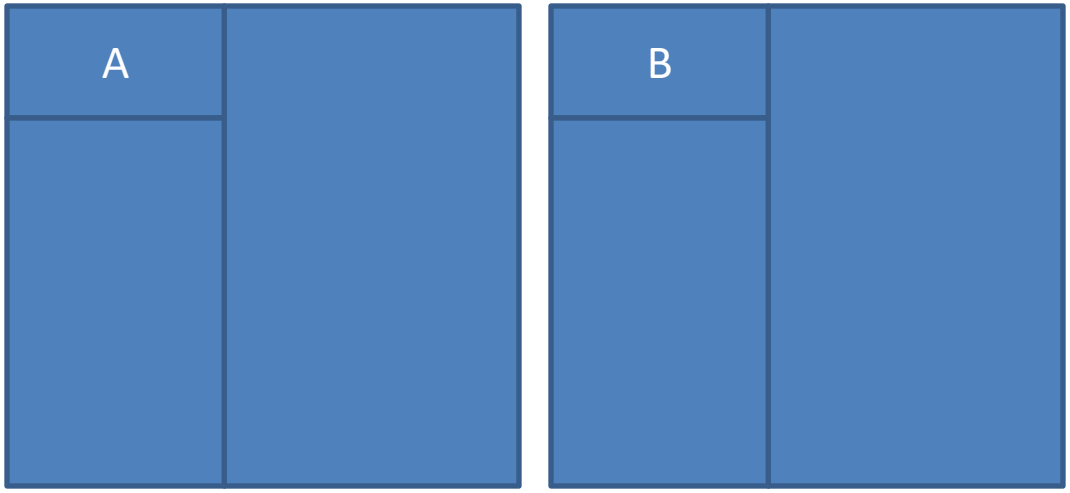
Write CRC Cards for Box Class

Box	Collaborators
Responsibilities 1. Getting length 2. Getting width 3. Getting height 4. Computing area and volume	<<None>>

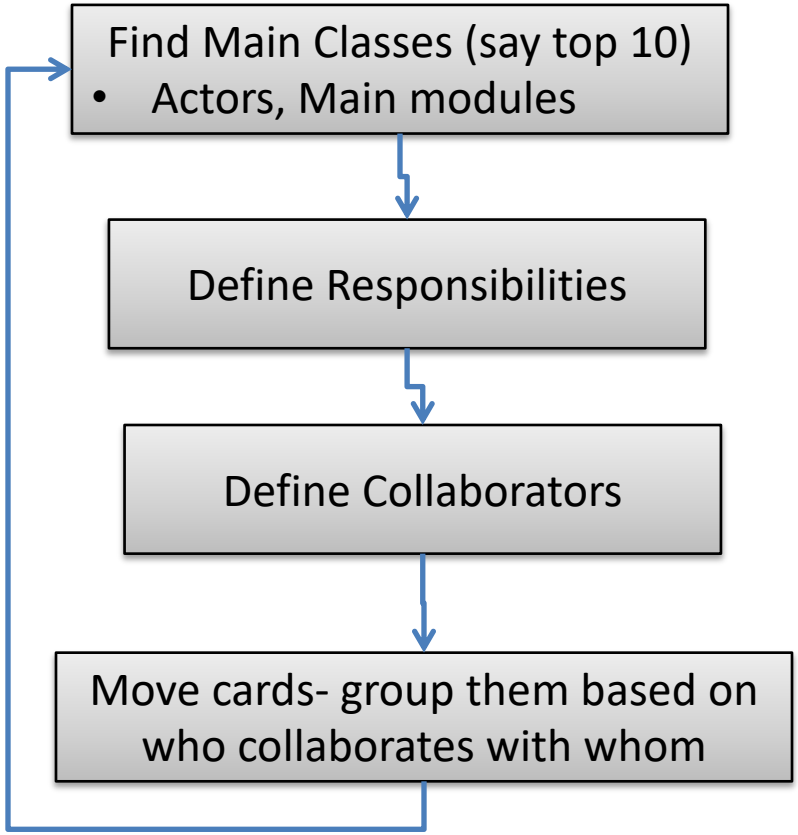
CRC Card Example 2

```
class B
{
  public void doB()
  {
    System.out.println("Hello");
  }
}
class A
{
  public void doS()
  {
      B b1 = new B();
      b1.doB();
  }
} //End of class Test
```

Write CRC Cards for Classes A & B



How do you create CRC Model?



Thank You

SS ZG653 (RL 8.3): Software Architecture

Introduction to Unified Process

Instructor: Prof. Santonu Sarkar

BITS Pilani

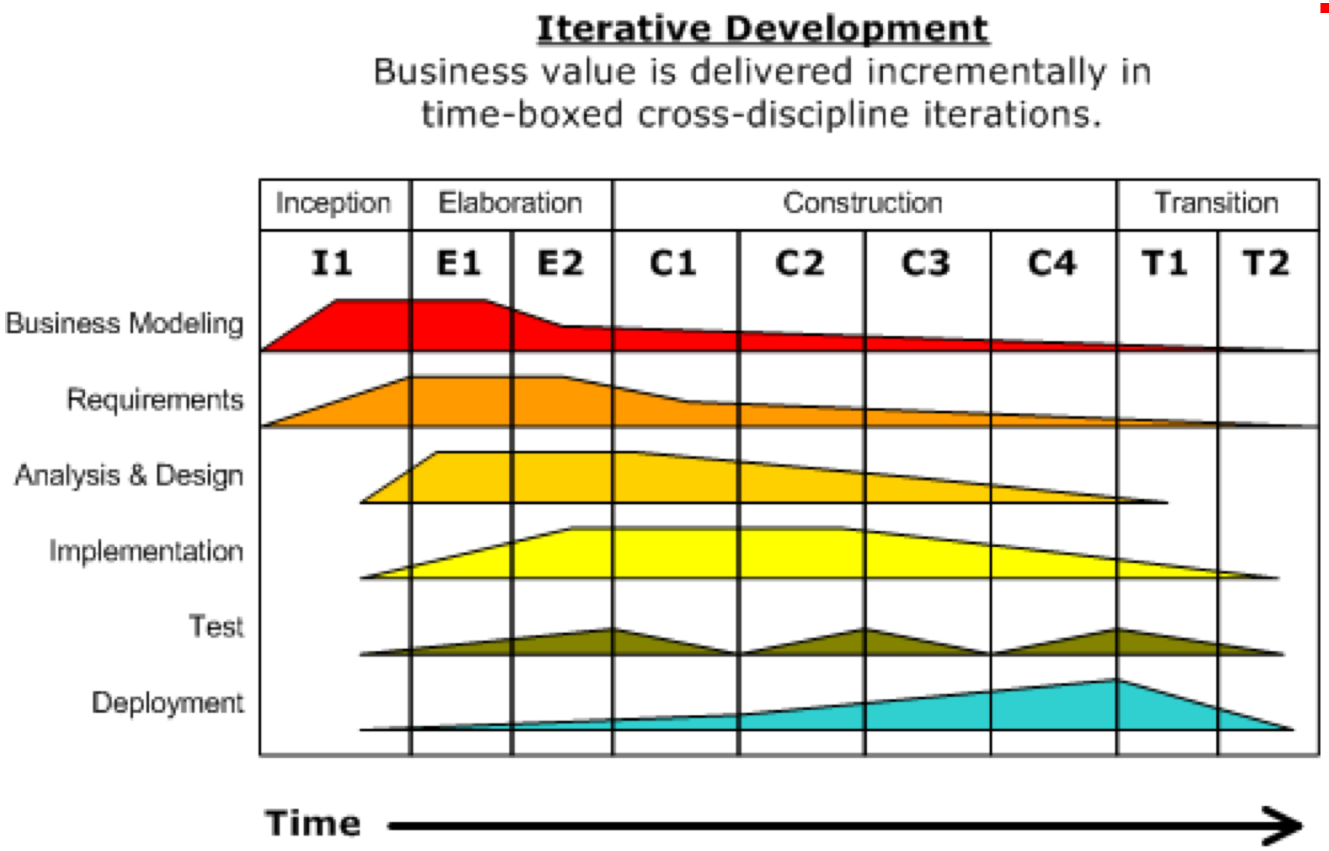
Pilani|Dubai|Goa|Hyderabad

What is (Rational) Unified Process

- While UML provided the necessary technology for OO software design
- Unified process gives a framework to build the software using UML
- Iterative approach
- Five main phases
  - Inception
    - Establish a justification and define project scope
    - Outline the use cases and key requirements that will drive the design tradeoffs Outline one or more candidate architectures
    - Identify risks and prepare a preliminary project schedule alongwith cost estimate
  - Elaboration (Architecture and Design)
  - Construction (Actual implementation)
  - Transition (initial release)
  - Production (Actual deployment)

– Rumbaugh, Booch, Jacobson

Architecture Activities



- Architecture Focused all the time
- Starts during inception and described in detail during the elaboration phase

Feature Driven Design

- Starts during Inception-Elaboration Phase
- Mostly used in the context of Agile development
- The requirement is modeled as a set of features
  - A feature is a client-valued functionality that can be implemented and demonstrated quickly
- 'Feature' template *<action>* **the** *<result>* (**by**|**for**|**of**|**to**) **a(n)** *<object>*
  - Add a product to a shopping-cart
  - Store the shipping-information for the customer

From Feature to Architecture

- Once a set of features are collected
- Similar features are grouped together into a module
- Set of clusters created out of the features, forms a set of modules
- This becomes the basic Module Structure
  - Recall the software architecture and views…

Use-case Model

- Use-cases are used to describe an usage scenario from the user's point of view
  - Create a basic use-case diagram
  - Elaborate each use-case
- To create Use-Case
  - Define actors (who will use this use-case)
  - Describe the scenario
    - Preconditions
    - Main tasks
    - Exceptions
    - Variation in the actors interaction
    - What system information will the actor acquire, produce or change?
    - Will the actor have to inform about the changes?
    - Does the actor wish to be informed about any unexpected changes?

First step towards Module Views

- Analysis Classes are not the final implementation level classes. They are more coarse grained. They are typically modules. They manifest as
  - External Entities
    - Other systems, devices that produce or consume information related to this system
  - Things
    - Report, letters, signals
  - Structure
    - Sensors, four-wheeled car,… that define a class of objects

Analysis Classes manifest as…

- Event Occurrences
  - Transfer of fund, Completion of Job
- Roles
  - People who interact with the systems (Manager, engineer, etc.)-- Actors
- Organizational Units
  - Divisions or groups that are important for this system
- Places
  - Location that establish the context of the overall functionality of the system

How to get these classes?

- Get the noun phrases. They are the potential objects.
- Apply the following heuristics to get a legitimate analysis class
  - It should retain information for processing
  - It should have a set of identifiable operations
  - Multiple attributes should be there for a class
  - Operations should be applicable for all instances of this class
  - Attributes should be meaningful for all instances of this class

Thank You