

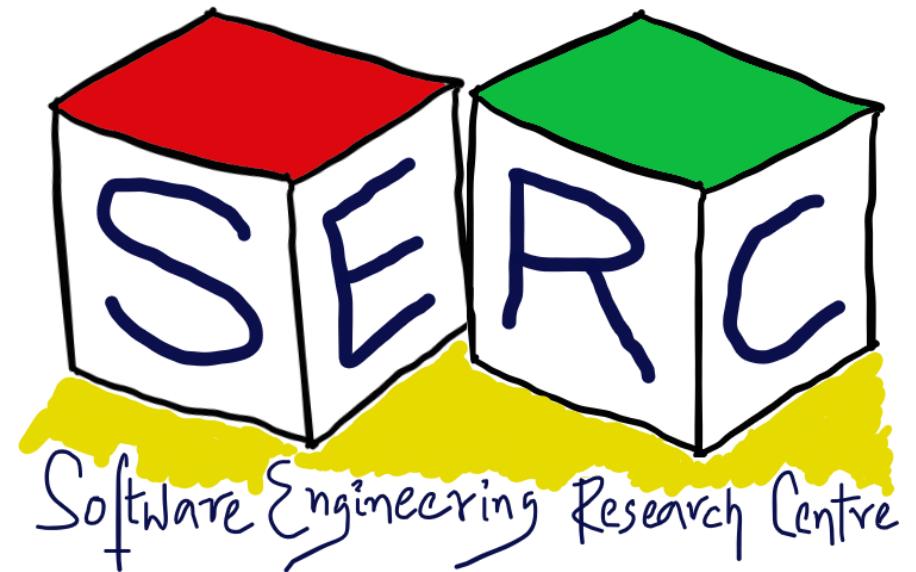
# Software Modeling: An Overview

**CS6.401 Software Engineering**

Dr. Karthik Vaidhyanathan

[karthik.vaidhyanathan@iiit.ac.in](mailto:karthik.vaidhyanathan@iiit.ac.in)

<https://karthikvaidhyanathan.com>



# Acknowledgements

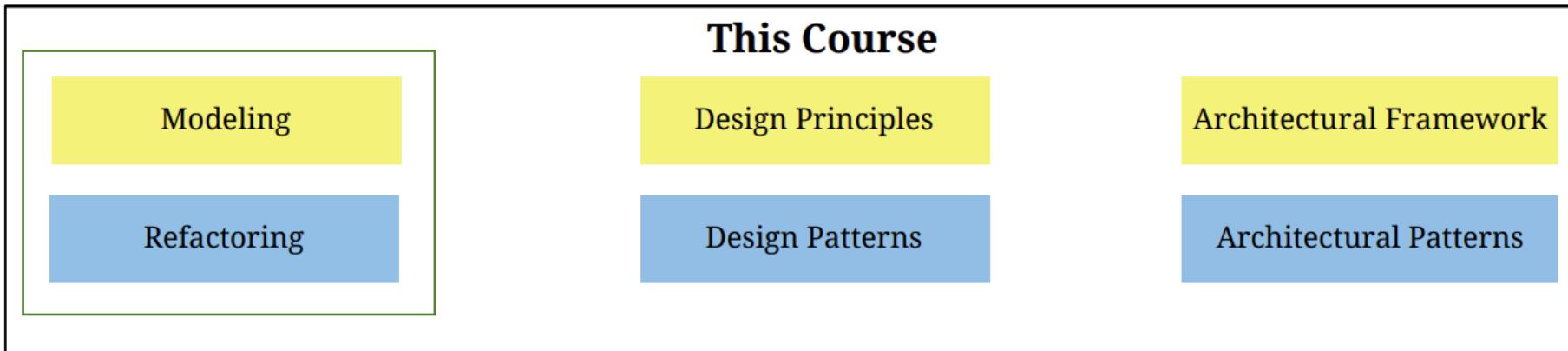
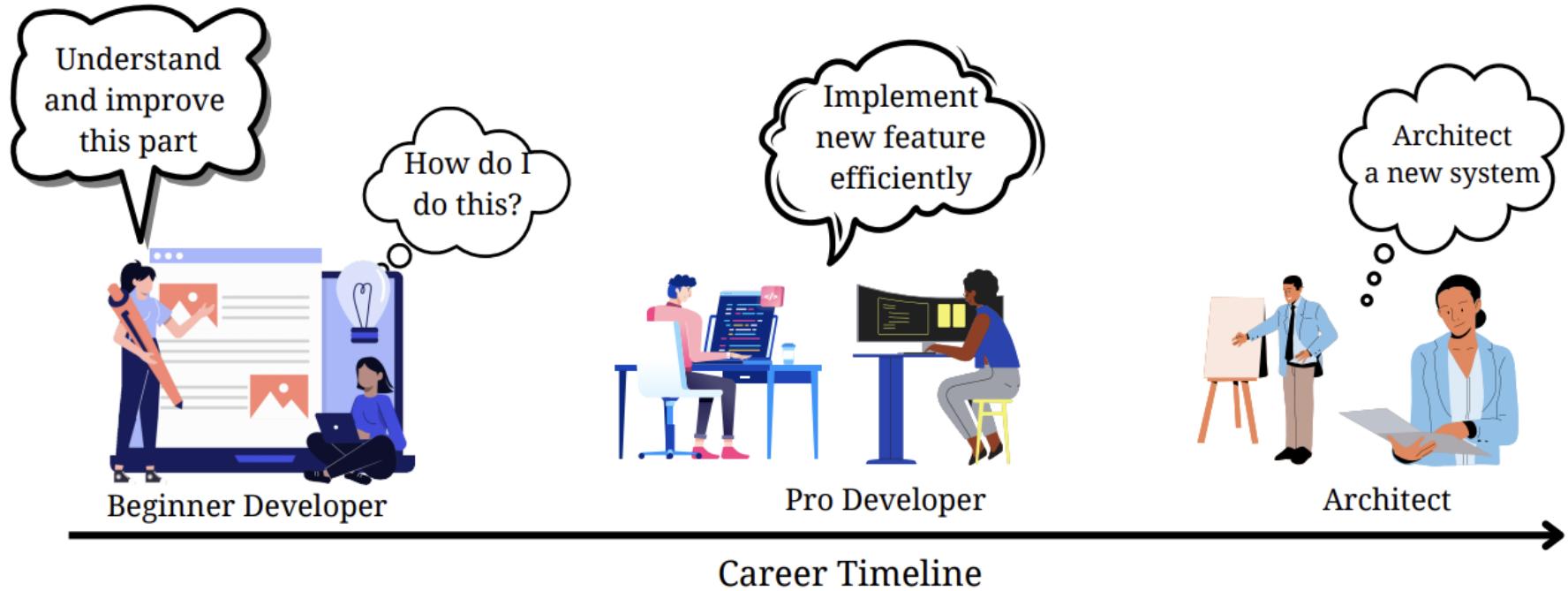
The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge

-- Karthik Vaidhyanathan

## Sources:

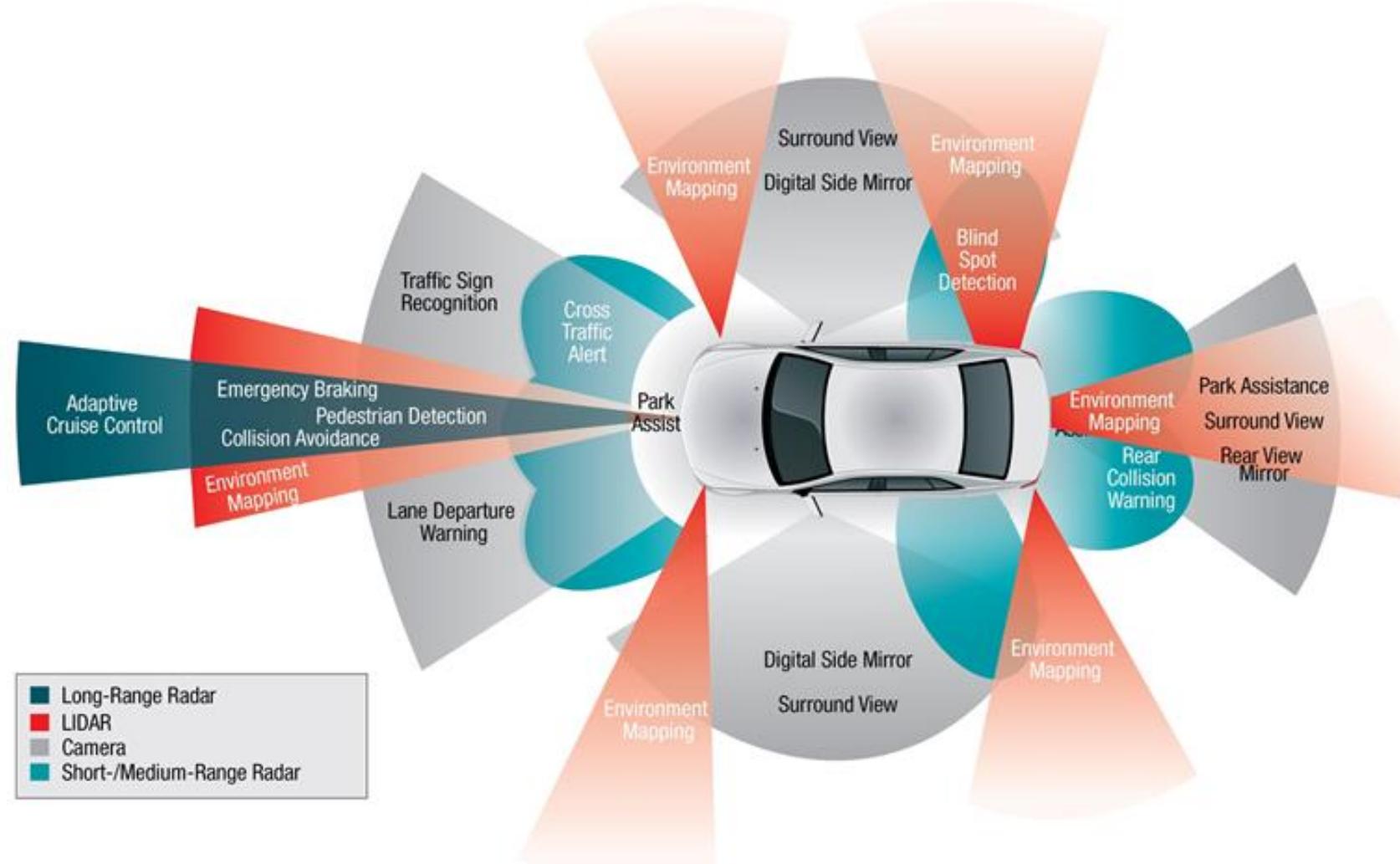
1. Introduction to MDE, Ludovico Iovino, GSSI, Italy
2. UML@Classroom, An Introduction to Object-Oriented Modeling by Martina Seidl, Marion Scholz, Christian Huemer and Gerti Kappel
3. UML Modelling lecture, Dr. Raghu, IIIT Hyderabad

# Course Outline



# What is a Model?

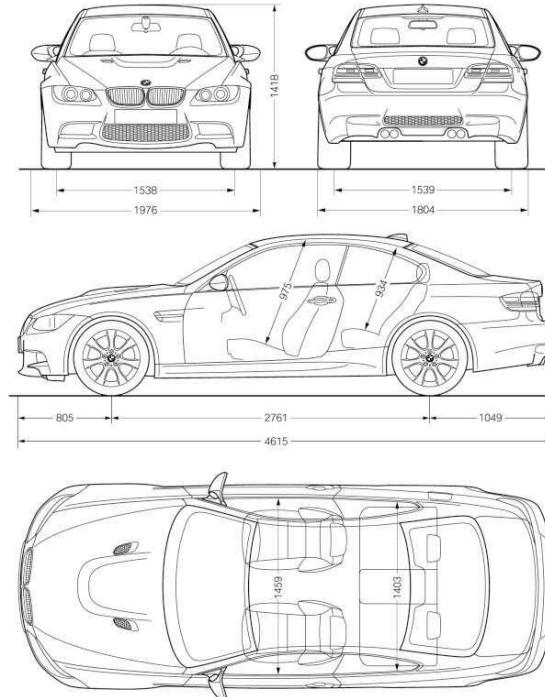
# Let us consider a real system



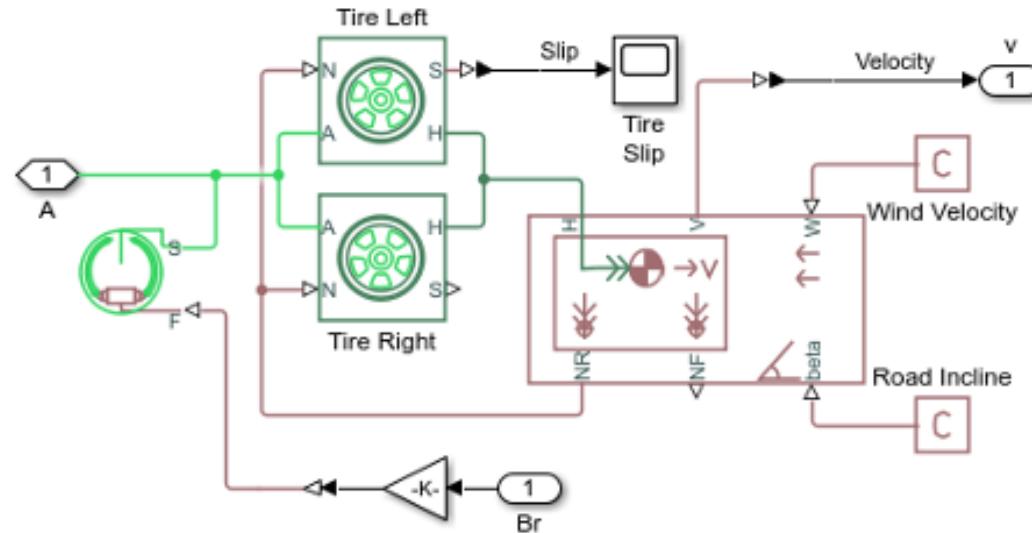
# Let us now consider a model of the system

*"The brain does much more than recollect. It compares, synthesizes, analyzes, generates abstractions."*

-- Carl Sagan



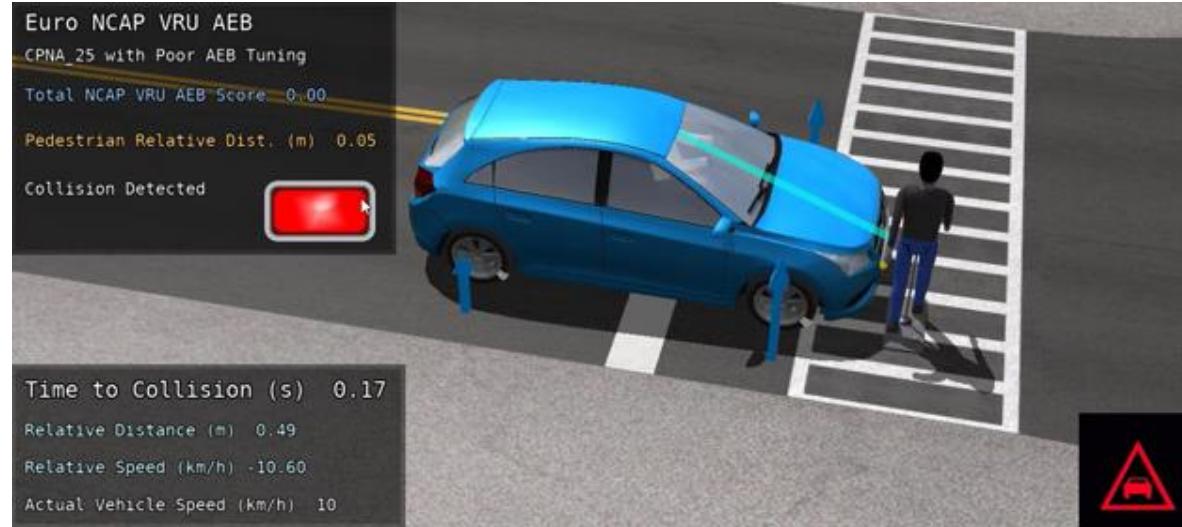
Model from the designers



Model from the engineers

Model is a simplification of a reality. In other words, it is a blueprint of the system

# Modelling can serve more purpose



Checking collision avoidance

Checking if the traffic signs are followed

In essence models can be simulated with tools to perform analysis

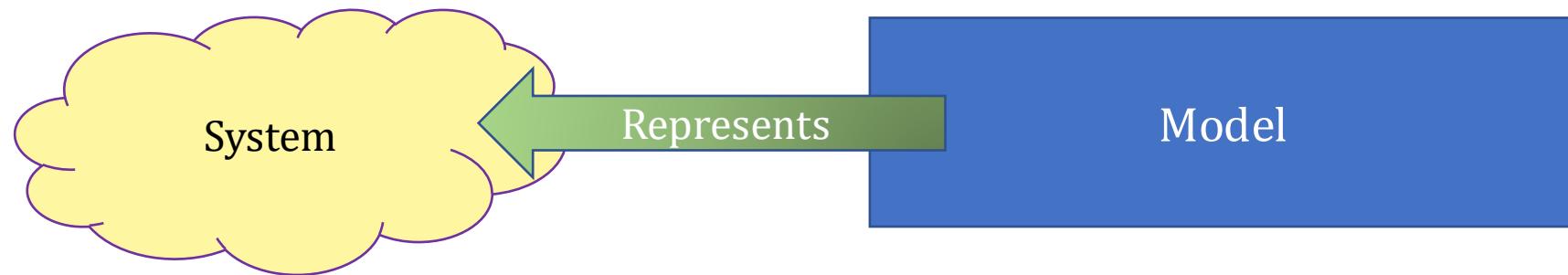


# Simulation using SUMO



# So what is a software model?

A simplified or partial representation of a real system, defined in order to **accomplish a task** or to **reach an agreement**.



**Mapping:** A model is always a mapping of some real system

**Reduction:** A model reflects only relevant set of properties of original system

**Pragmatism:** A model needs to be usable in place of the actual system with respect to some purpose

# Quality of Model

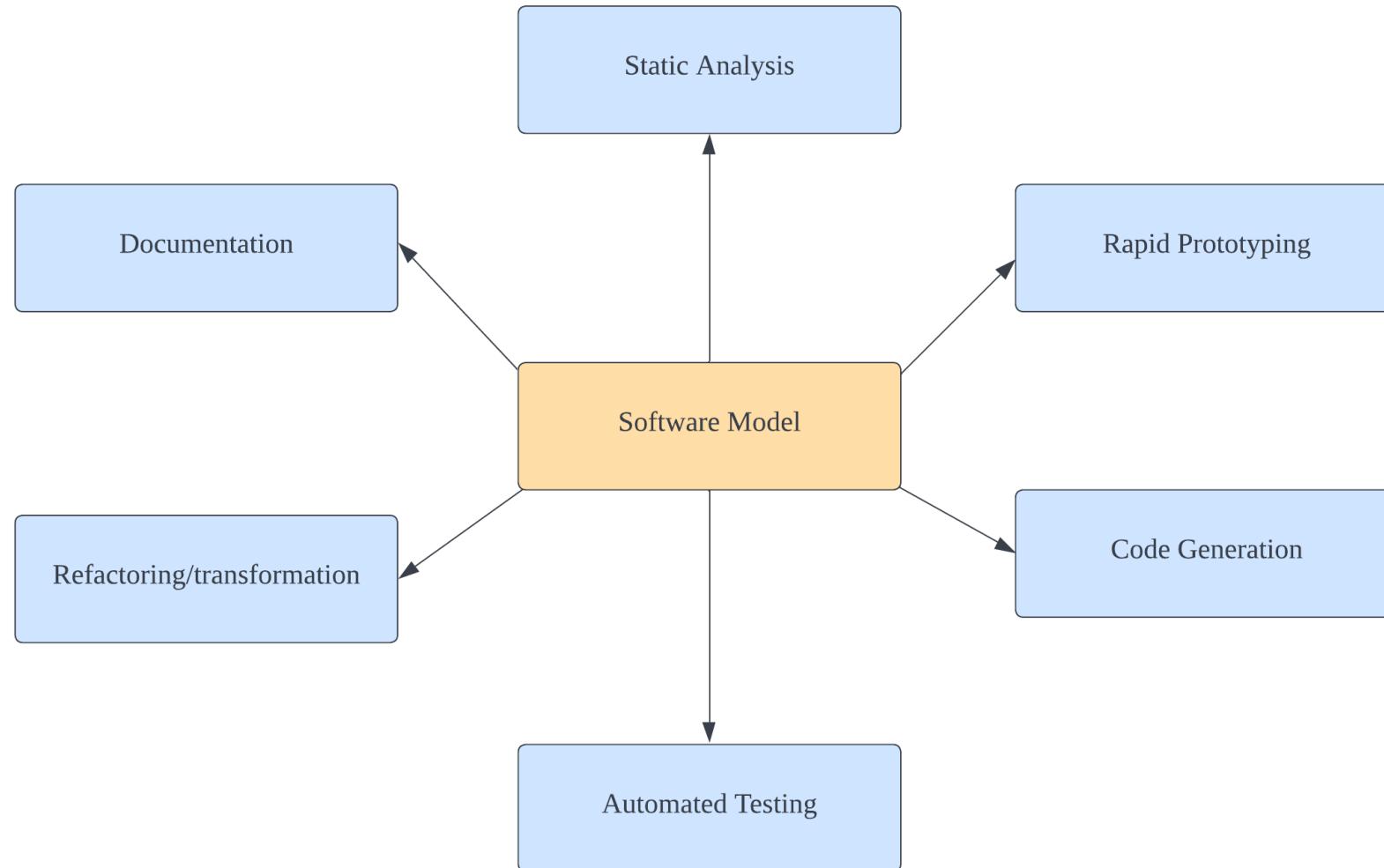
As per Bran Selic, five characteristics determine model's quality

- Abstraction: A model should be reduced version of system [Omit unwanted]
- Understandability: Should be as intuitive as possible
- Accuracy: Reflect relevant properties as close to reality as possible
- Predictiveness: Enable prediction of interesting properties of system
- Cost-effectiveness: Cheaper to create models than the system

# Glimpse into world of Model-driven Engineering

# Model Driven Software Engineering

Shifting focus from code centric techniques to models



# What is MDE? – Key Motivation

## Models as a sketch

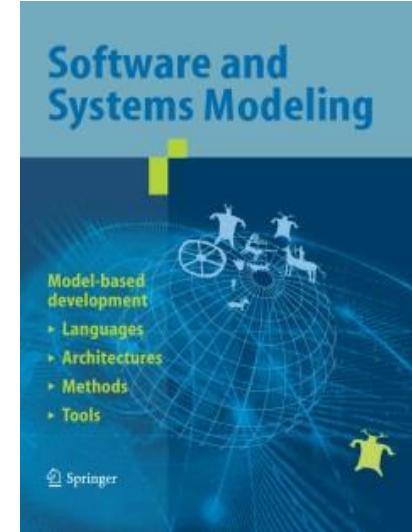
- Communication of ideas
- Objective: Modeling per se

## Models as guidelines/blueprint

- Design decisions are documented
- Objective: instrumentation for implementation

## Models as executable programs

- Generate code automatically
- Objective: models are source code and vice versa



Sosym journal



Models conference



# What to Model?

# Multiple ways to think about it

## Algorithmic Perspective

- Main block of building the software is procedure or function
- Scale and new features affects maintainability and reasoning

## Object oriented Perspective

- Main building block of all software system is object or class
- Contemporary view of software development

# Object Oriented Modeling

- Model system as a collection of objects that interact with each other
- Tries to captures the real-world scenario
  - Everything is an object
  - Has a state and behavior: (Happy, angry)...(speaking softly, yelling...)
  - Can you model a person?
- Software objects are similar to real world objects:
  - Store state in fields (variables)
  - Behavior through methods (functions)

# Objects vs Classes

	<b>Real world</b>	<b>Modeling World</b>
Object	Object represents anything that can be distinctly identified	An object has identity, state and behavior
Class	Represents set of objects with similar characteristics and behavior	It characterizes the structure of states and behaviors shared by its instances.

Object is like a variable of a class

# Lets take a scenario – E-commerce System, Lets think!



**What users see**



**Organization/admins**



**Developers building system**



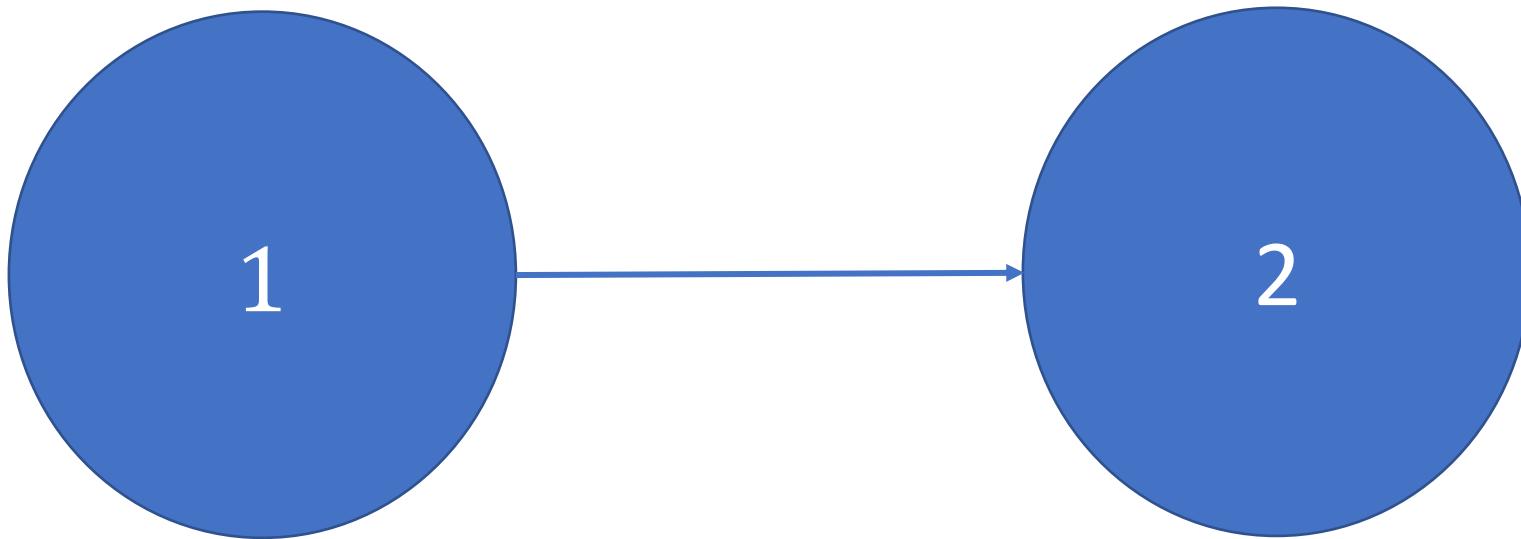
**Behind the scenes!**

# Key Characteristics

- Abstraction: Hide irrelevant details (eg: Coffee machine)
  - Deals with What!
- Encapsulation: Protection against unauthorized access (eg: Organization)
  - Deals with how!
- Relationships
  - Inheritance: Derive classes from existing classes (eg: Real life inheritance!!)
  - Association: Relation between two classes (Aggregation, composition)
  - Dependency: Some form of dependency between two classes

# How to Model?

# How do you interpret this?



- 2 comes after 1
- 2 depends on 1
- 1 specializes/refines 2
- 2 listens to 1
- 1 contains 2

...

# Modeling Languages

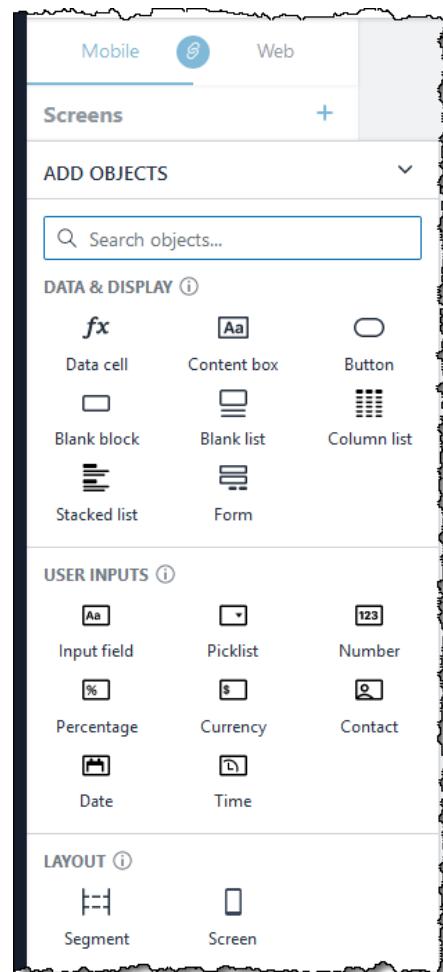
# Modeling Languages

Largely classified into two types

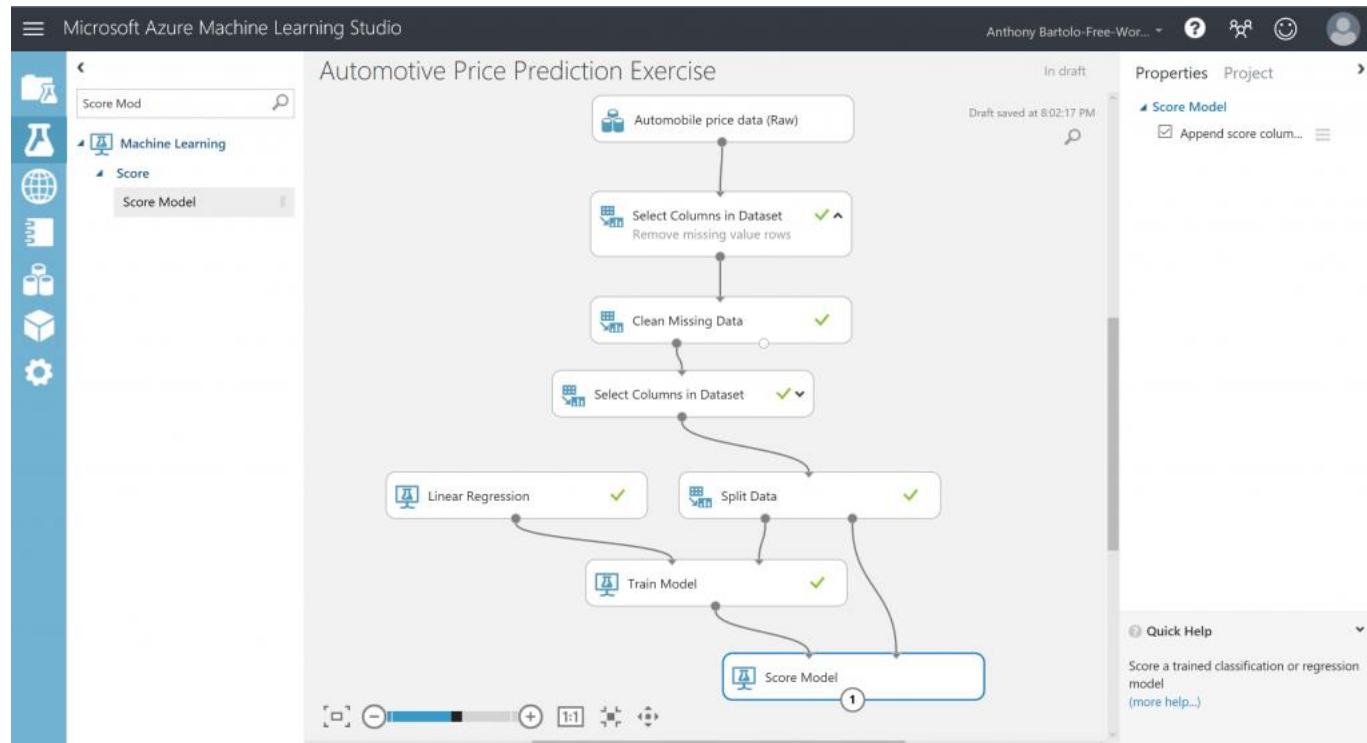
- Domain-Specific Languages (DSLs)
  - Languages designed to model a certain domain
  - Examples: HTML, SQL, etc.
- General Purpose Modeling Languages (GPLs)
  - Languages can be applied to any domain for modeling
  - Examples: UML, XML, etc.

Wait is this similar to low code/no code?

# Some Examples

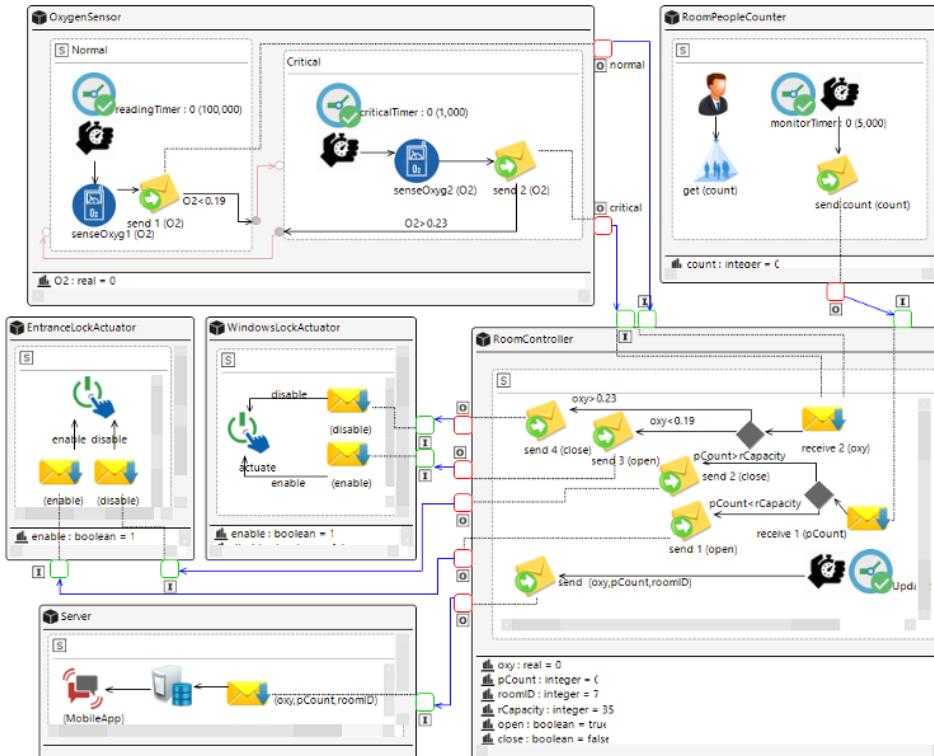


AWS Honeycode



Azure Machine Learning Studio

# Some Examples - We can create our own too

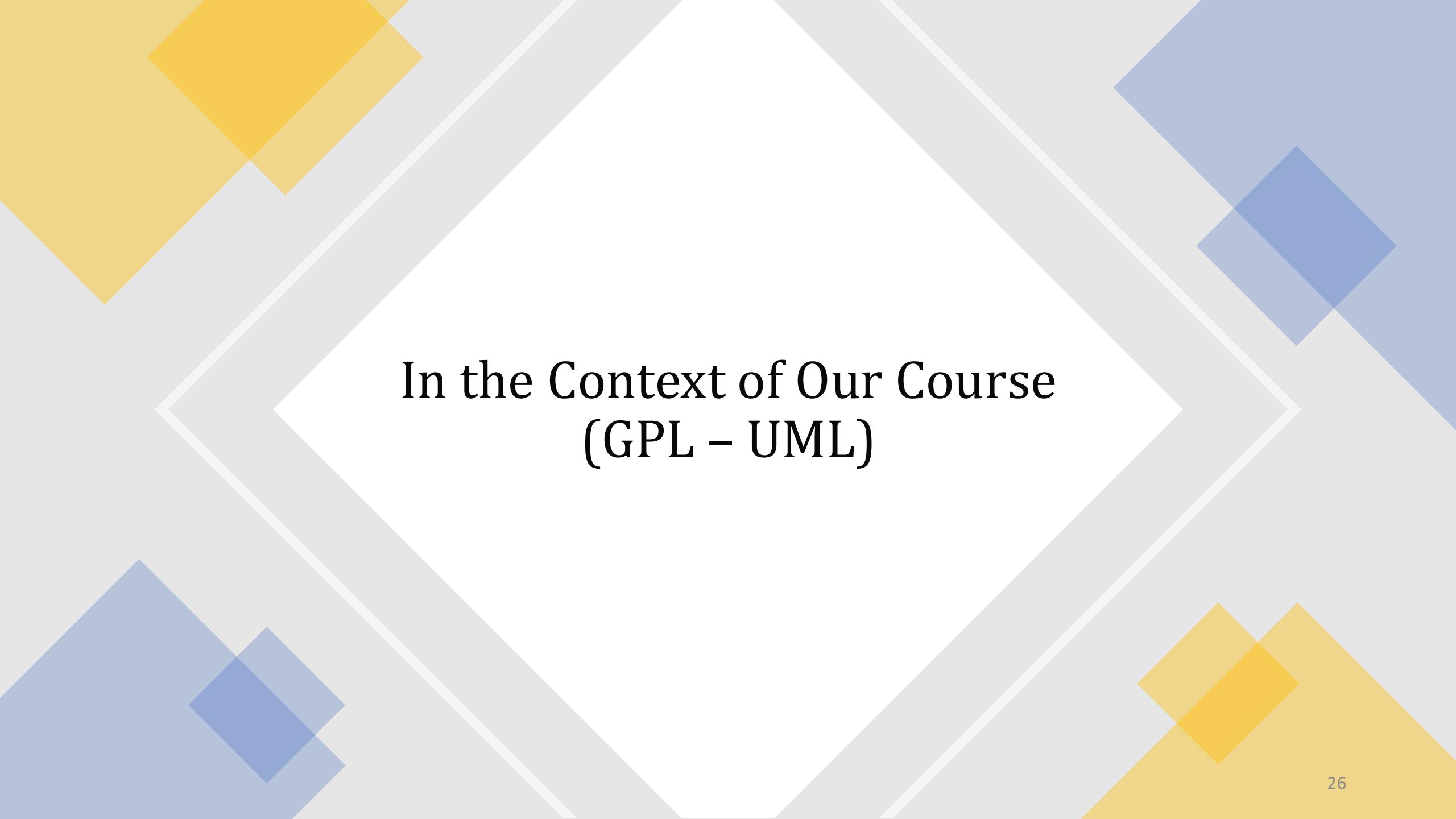


CAPS modeling for IoT

**configuration** Robot

```
{  
    instance robot : RobotControl  
    instance sdist : DistanceSensor  
    instance scoll : CollisionSensor  
    instance motion : MotionControl  
    instance left_wheel : WheelControl  
    instance right_wheel : WheelControl  
  
    connector robot.rangefinder => sdist.data  
    connector robot.bumper => scoll.data  
    connector robot.platform => motion.ctrl  
    connector motion.left => left_wheel.ctrl  
    connector motion.right => right_wheel.ctrl  
}
```

ThingML modeling



In the Context of Our Course  
(GPL – UML)

# Unified Modeling Language (UML): Brief History

- No common language to model until 1996
- GPL developed by industry consortium in 1997
  - Introduction of OOP in IT dates back to 1960's
  - Required a standard representation: **OMG**
  - Three Amigos: Grady Booch, Ivar Jacobson and James Rumbaugh
- Based on multiple prior visual modeling languages
- Goal was to have a single language that could cover large number of SE tasks
- Current version of UML: 2.5.1 (as of Dec 2017)



<https://www.omg.org/spec/UML/>

# Unified Modeling Language (UML)

- Notation for OO Modeling
  - Use object orientation as basis
  - Model a system as collection of objects that interact with each other
- Graphical diagrams as a way to model systems
  - More clear (imprecise) than natural language (too detailed)
  - Capture an overall view of the system
  - Independent of language or technology

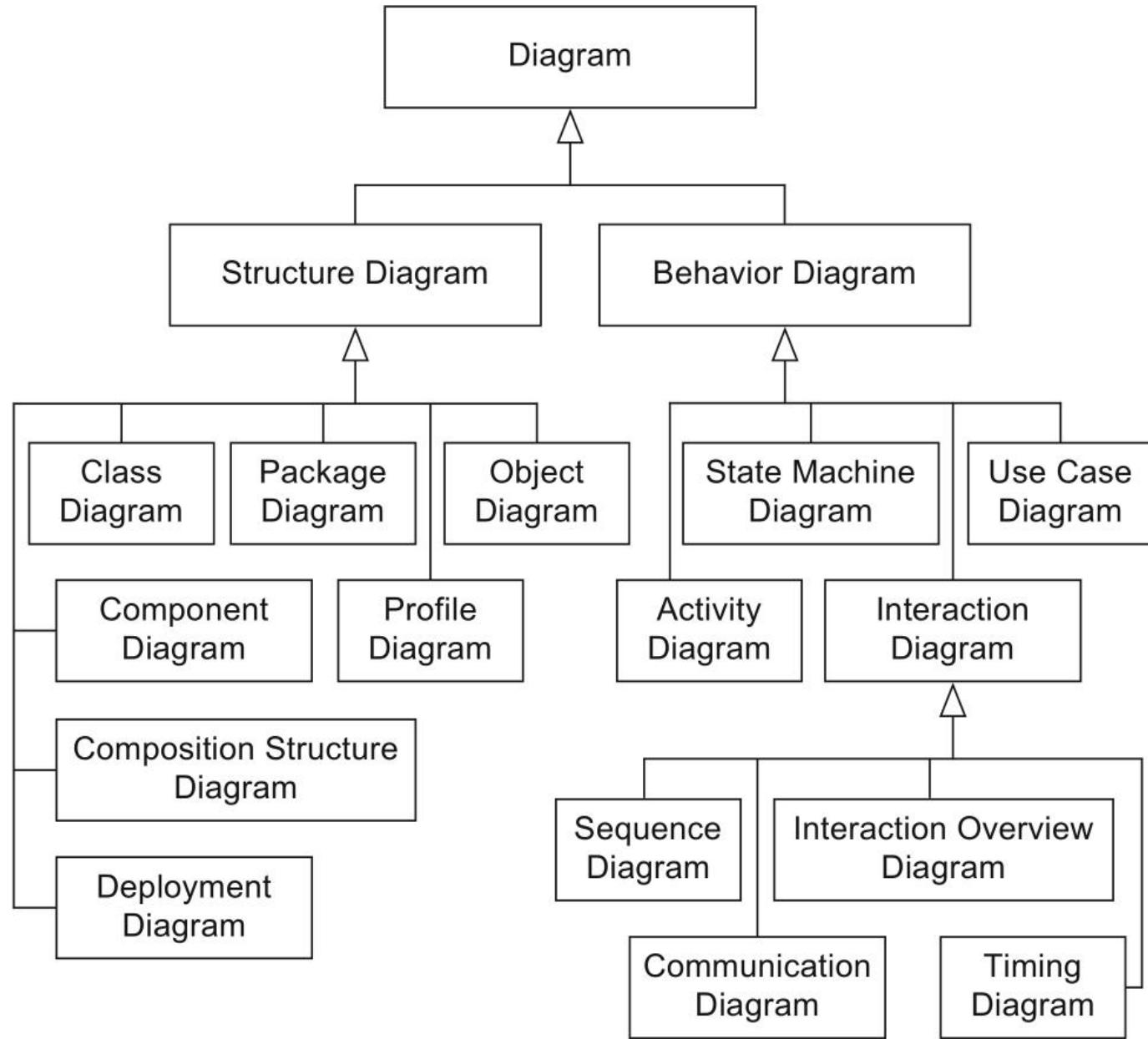
# What UML is not?

- Not an OO Method or Process
- Not a visual programming language
- Not a tool specification



# UML Diagrams

- 14 different diagrams
- Structure diagrams for capturing static aspects of system
- Behavior diagrams for capturing dynamic aspect of system



# Can you create a model?

Think of a course management system like moodle. Can you create a model for the same?

Just use whatever knowledge you have, any type of diagram as per your knowledge is fine – Give a try!!



# Thank You



Course website: [karthikv1392.github.io/cs6401\\_se](https://karthikvaidhyanathan.github.io/cs6401_se)

Email: [karthik.vaidhyanathan@iiit.ac.in](mailto:karthik.vaidhyanathan@iiit.ac.in)  
Web: <https://karthikvaidhyanathan.com>  
Twitter: @karthi\_ishere