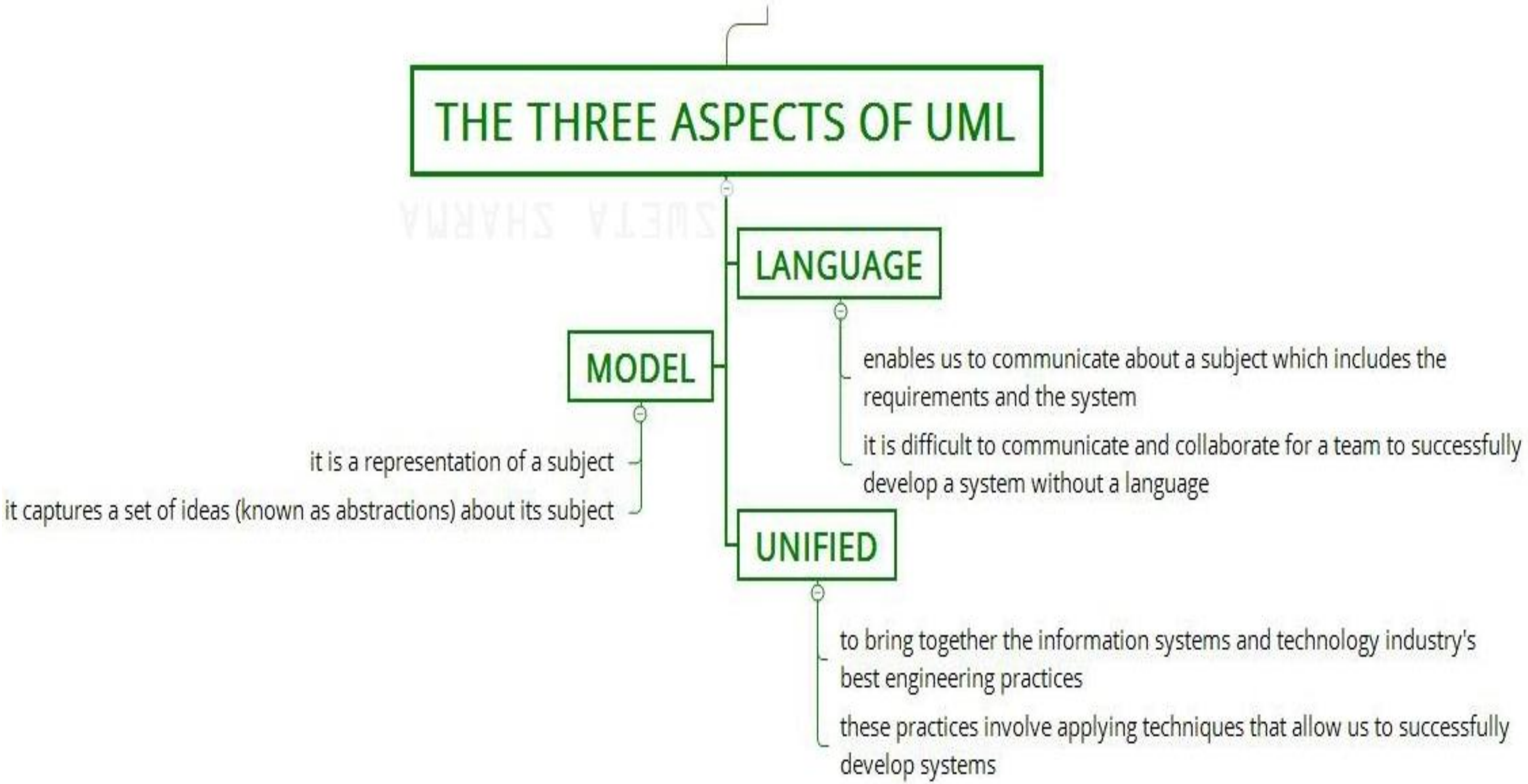
Conceptual Model of the Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a standard visual language for describing and modelling software blueprints. The UML is more than just a graphical language.

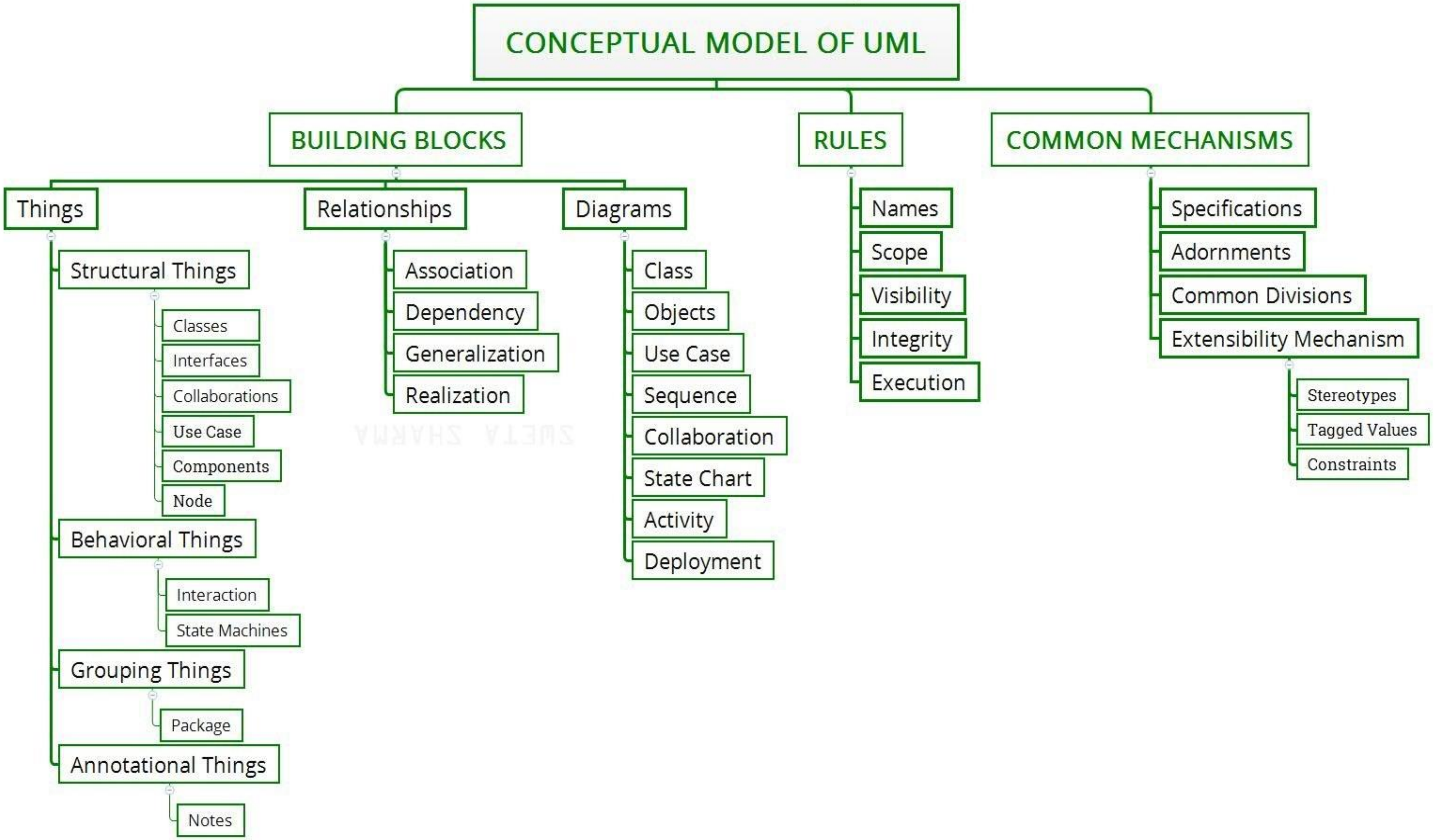
Stated formally, the UML is for: Visualizing, Specifying, Constructing, and Documenting.

The artifacts of a software-intensive system (particularly systems built using the object-oriented style).



THE THREE ASPECTS OF UML

- MODEL
  - it is a representation of a subject
  - it captures a set of ideas (known as abstractions) about its subject
- LANGUAGE
  - enables us to communicate about a subject which includes the requirements and the system
  - it is difficult to communicate and collaborate for a team to successfully develop a system without a language
- UNIFIED
  - to bring together the information systems and technology industry's best engineering practices
  - these practices involve applying techniques that allow us to successfully develop systems


CONCEPTUAL MODEL OF UML
BUILDING BLOCKS
Things
Structural Things
Classes
Interfaces
Collaborations
Use Case
Components
Node
Behavioral Things
Interaction
State Machines
Grouping Things
Package
Annotational Things
Notes
Relationships
Association
Dependency
Generalization
Realization
Diagrams
Class
Objects
Use Case
Sequence
Collaboration
State Chart
Activity
Deployment
RULES
Names
Scope
Visibility
Integrity
Execution
COMMON MECHANISMS
Specifications
Adornments
Common Divisions
Extensibility Mechanism
Stereotypes
Tagged Values
Constraints

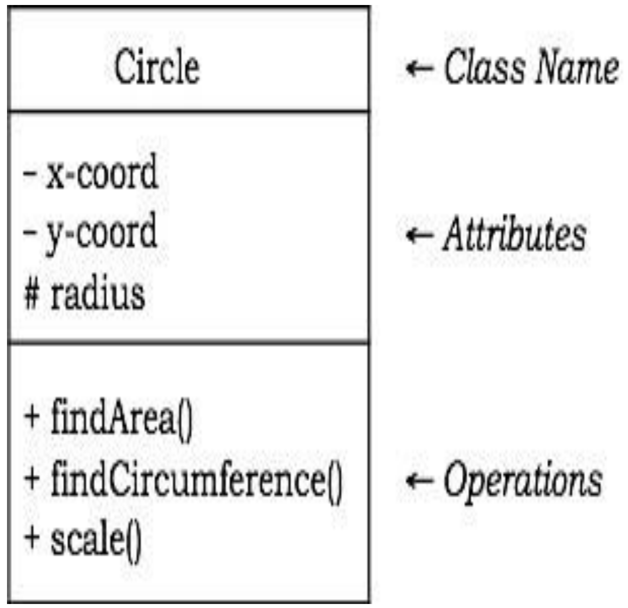
Class

- A class is represented by a rectangle having three sections −
- the top section containing the name of the class
- the middle section containing class attributes
- the bottom section representing operations of the class

The visibility of the attributes and operations can be represented in the following ways −

- **Public** − A public member is visible from anywhere in the system. In class diagram, it is prefixed by the symbol ‘+’.
- **Private** − A private member is visible only from within the class. It cannot be accessed from outside the class. A private member is prefixed by the symbol ‘−’.
- **Protected** − A protected member is visible from within the class and from the subclasses inherited from this class, but not from outside. It is prefixed by the symbol ‘#’.
- An abstract class has the class name written in italics.

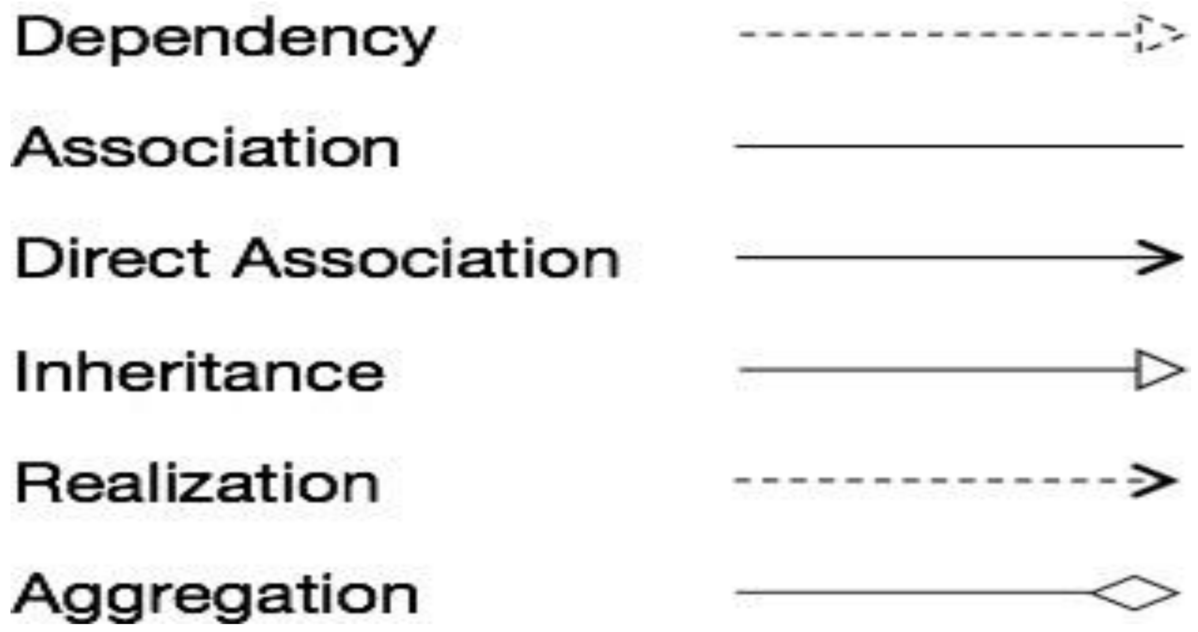
- **Example** − Let us consider the Circle class introduced earlier. The attributes of Circle are x-coord, y-coord, and radius. The operations are findArea(), findCircumference(), and scale(). Let us assume that x-coord and y-coord are private data members, radius is a protected data member, and the member functions are public. The following figure gives the diagrammatic representation of the class.



Relationship

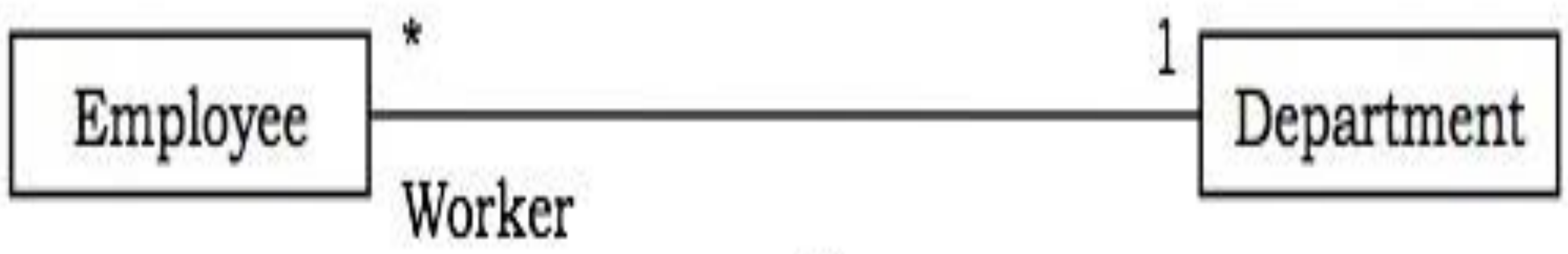
The notations for the different types of relationships are as follows

- Relationship
- The notations for the different types of relationships are as follows

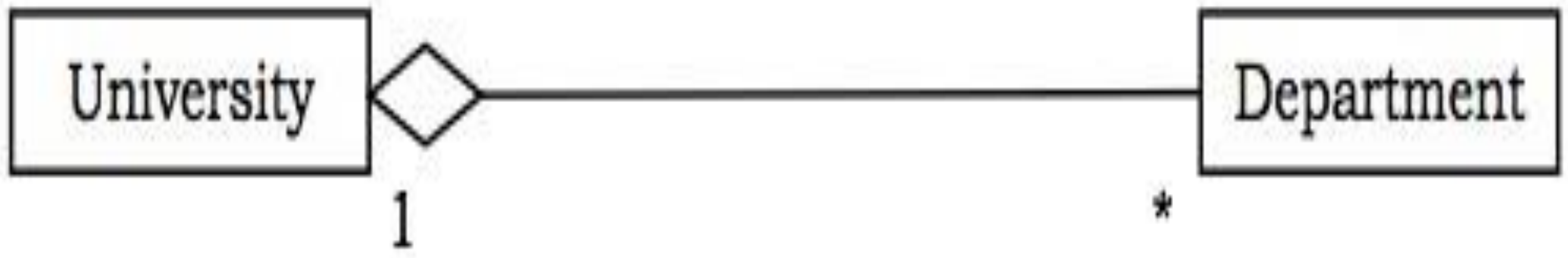


- Usually, elements in a relationship play specific roles in the relationship. A role name signifies the behavior of an element participating in a certain context.

- **Example** − The following figures show examples of different relationships between classes. The first figure shows an association between two classes, Department and Employee, wherein a department may have a number of employees working in it. Worker is the role name. The ‘1’ alongside Department and ‘*’ alongside Employee depict that the cardinality ratio is one–to–many. The second figure portrays the aggregation relationship, a University is the “whole–of” many Departments.



(a)



(b)

Common Mechanisms……

UML has four common mechanisms −

1. Specifications
2. Adornments
3. Common Divisions
4. Extensibility Mechanisms

- **Specifications**

In UML, behind each graphical notation, there is a textual statement denoting the syntax and semantics. These are the specifications. The specifications provide a semantic backplane that contains all the parts of a system and the relationship among the different paths.

Adornments

Each element in UML has a unique graphical notation. Besides, there are notations to represent the important aspects of an element like name, scope, visibility, etc.

Common Divisions

- Object-oriented systems can be divided in many ways. The two common ways of division are −
- **Division of classes and objects** − A class is an abstraction of a group of similar objects. An object is the concrete instance that has actual existence in the system.
- **Division of Interface and Implementation** − An interface defines the rules for interaction. Implementation is the concrete realization of the rules defined in the interface.

Extensibility Mechanisms

- UML is an open-ended language. It is possible to extend the capabilities of UML in a controlled manner to suit the requirements of a system. The extensibility mechanisms are −
- **Stereotypes** − It extends the vocabulary of the UML, through which new building blocks can be created out of existing ones.
- **Tagged Values** − It extends the properties of UML building blocks.
- **Constraints** − It extends the semantics of UML building blocks.

IMPORTANCE OF UML

A picture is worth a thousand words, this idiom absolutely fits describing UML. Object-oriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the object-oriented development. It was then that UML came into picture.

- There are a number of goals for developing UML but the most important is to define some general purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use.

- UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system.
- In conclusion, the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today’s complex environment.

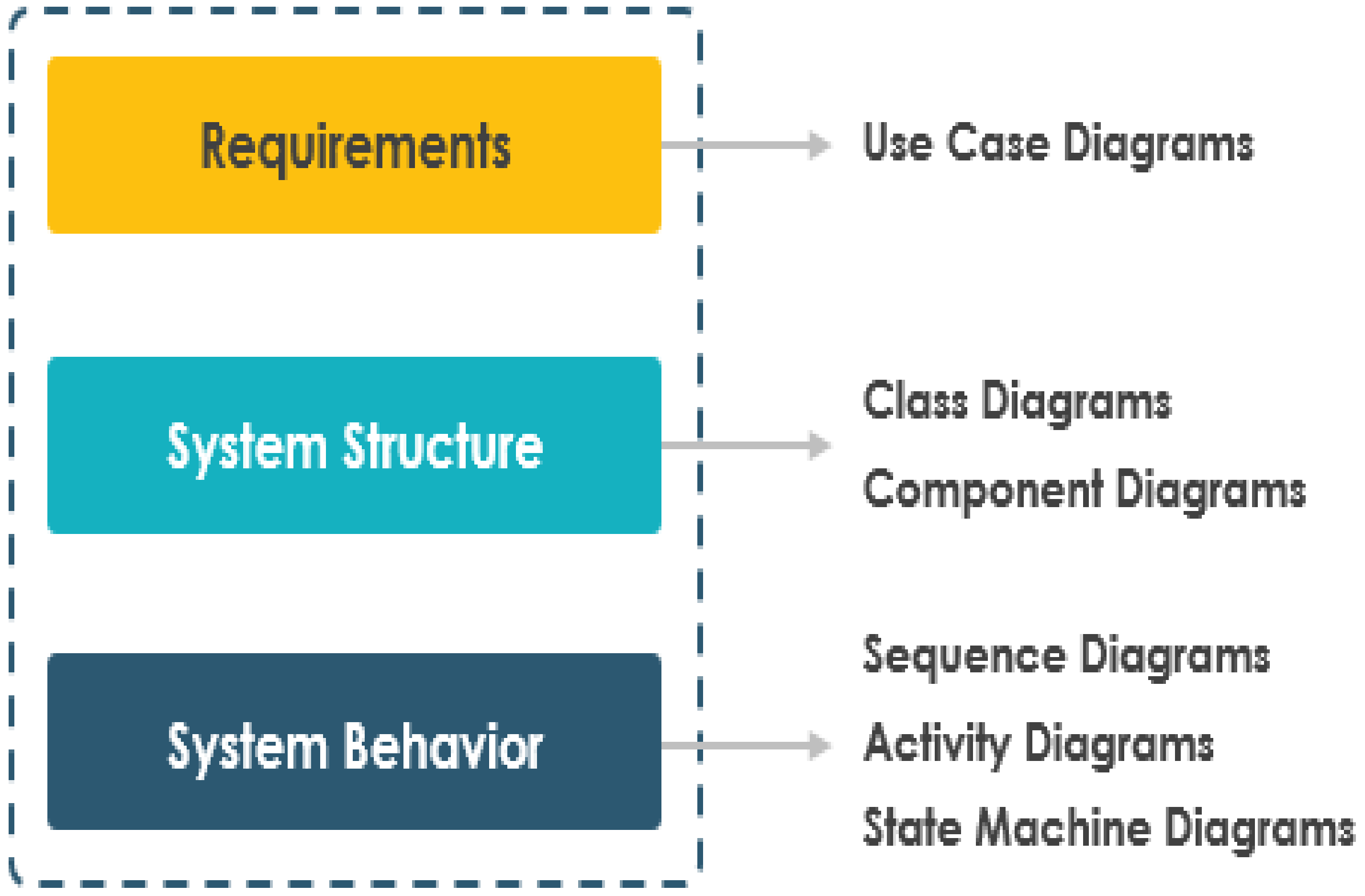
Principles of UML Modeling

1. The choice of model is important

The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped. We need to choose your models well.

•The right models will highlight the most critical development problems.

•Wrong models will mislead you, causing you to focus on irrelevant issues.


Requirements
System Structure
System Behavior
Use Case Diagrams
Class Diagrams
Component Diagrams
Sequence Diagrams
Activity Diagrams
State Machine Diagrams

- 2. Every model may be expressed at different levels of precision
- For Example,
- If you are building a high rise, sometimes you need a 30,000-foot view for instance, to help your investors visualize its look and feel.
- Other times, you need to get down to the level of the studs for instance, when there's a tricky pipe run or an unusual structural element.

- 3. The best models are connected to reality
- All models simplify reality and a good model reflects important key characteristics.

- 4. No single model is sufficient
- Every non-trivial system is best approached through a small set of nearly independent models. Create models that can be built and studied separately, but are still interrelated. In the case of a building: