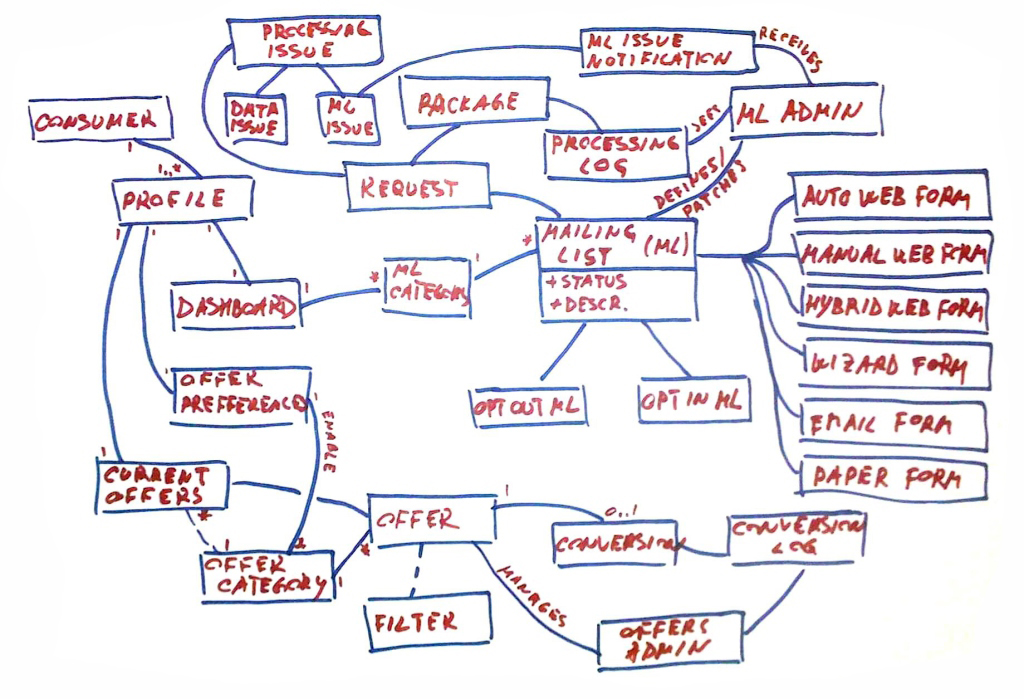
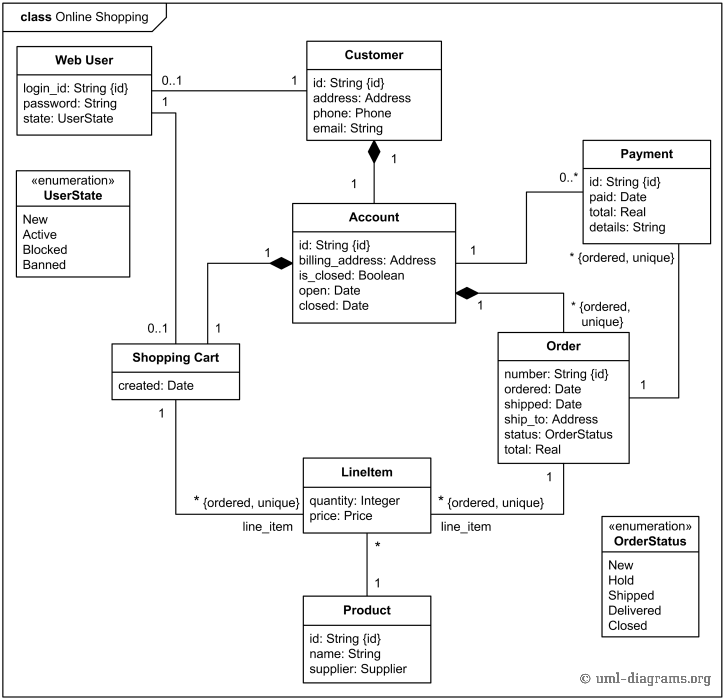
What is a domain model?

Domain Modeling is a way to describe and model real world entities and the relationships between them, which collectively describe the problem domain space. Derived from an understanding of system-level requirements, identifying domain entities and their relationships provides an effective basis for understanding and helps practitioners design systems for maintainability, testability, and incremental development. Because there is often a gap between understanding the problem domain and the interpretation of requirements, domain modeling is a primary modeling area in Agile development at scale. Driven in part from object-oriented design approaches, domain modeling envisions the solution as a set of domain objects that collaborate to fulfill system-level scenarios.  
  
**Agile software development** describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.These principles support the definition and continuing evolution of many software development methods.



Domain Model for Consumer Subscription Management System



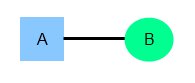
**ER MODEL:**

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of [data](http://www.webopedia.com/TERM/D/data.html) within [databases](http://www.webopedia.com/TERM/D/database.html) or information systems. An entity is a piece of data-an [object](http://www.webopedia.com/TERM/O/object.html)or concept about which data is stored.

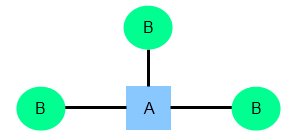
***Relationships Between Entities***

A relationship is how the data is shared between entities. There are three types of relationships between entities:

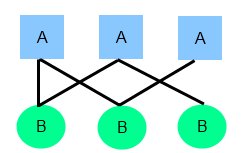
1. One to one

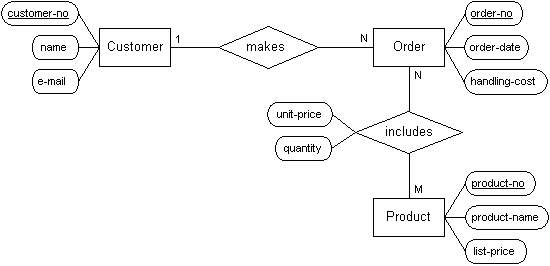


1. One to many



1. Many to many





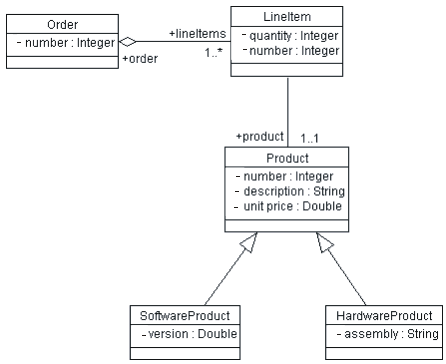
ER diagram for customer and product

**OBJECT MODEL**

An object model is a logical interface, software or system that is modeled through the use of object-oriented techniques. It enables the creation of an architectural software or system model prior to development or programming.An object model is part of the object-oriented programming (OOP) lifecycle.

An object model helps describe or define a software/system in terms of objects and classes. It defines the interfaces or interactions between different models, inheritance, encapsulation and other object-oriented interfaces and features.   
  
Object model examples include:

* **Document Object Model (DOM)**: A set of objects that provides a modeled representation of dynamic HTML and XHTML-based Web pages
* **Component Object Model (COM)**: A proprietary Microsoft software architecture used to create software components



Object Model diagram

UML:

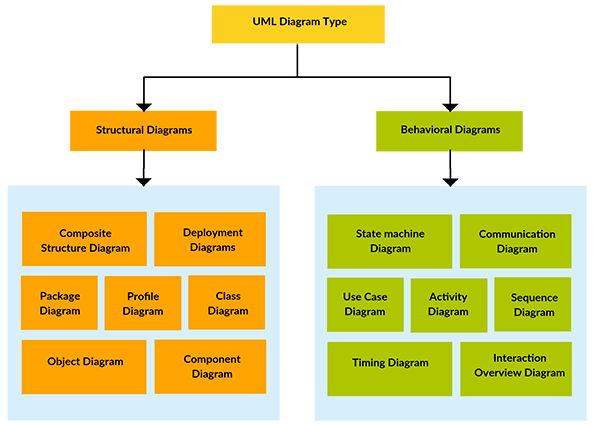
aUML stands for **U**nified **M**odeling **L**anguage. It’s a rich language to model software solutions, application structures, system behavior and business processes. There are **14 UML diagram types** to help you model these behaviors.

You can [**draw UML diagrams online**](http://creately.com/Draw-UML-and-Class-Diagrams-Online) using our software, or check out some UML diagram examples at our diagramming community.

### List of UML Diagram Types

So what are the different UML diagram types? There are two main categories; **structure diagrams** and **behavioral diagrams**. Click on the links to learn more about a specific diagram type.

* Structure Diagrams
  + [**Class Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#ClassDiagram)
  + [**Component Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#ComponentDiagram)
  + [**Deployment Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#DeploymentDiagram)
  + [**Object Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#ObjectDiagram)
  + [**Package Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#PackageDiagram)
  + [**Profile Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#ProfileDiagram)
  + [**Composite Structure Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#CompStrDiagram)
* Behavioral Diagrams
  + [**Use Case Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#UseCaseDiagram)
  + [**Activity Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#ActivityDiagram)
  + [**State Machine Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#StateMachDiagram)
  + [**Sequence Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#SequenceDiagram)
  + [**Communication Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#CommDiagram)
  + [**Interaction Overview Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#IntOverDiagram)
  + [**Timing Diagram**](http://creately.com/blog/diagrams/uml-diagram-types-examples/#TimingDiagram)

**[](http://static3.creately.com/blog/wp-content/uploads/2012/02/UML-Diagram-types1.png)**

*UML Diagram Types*