Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak

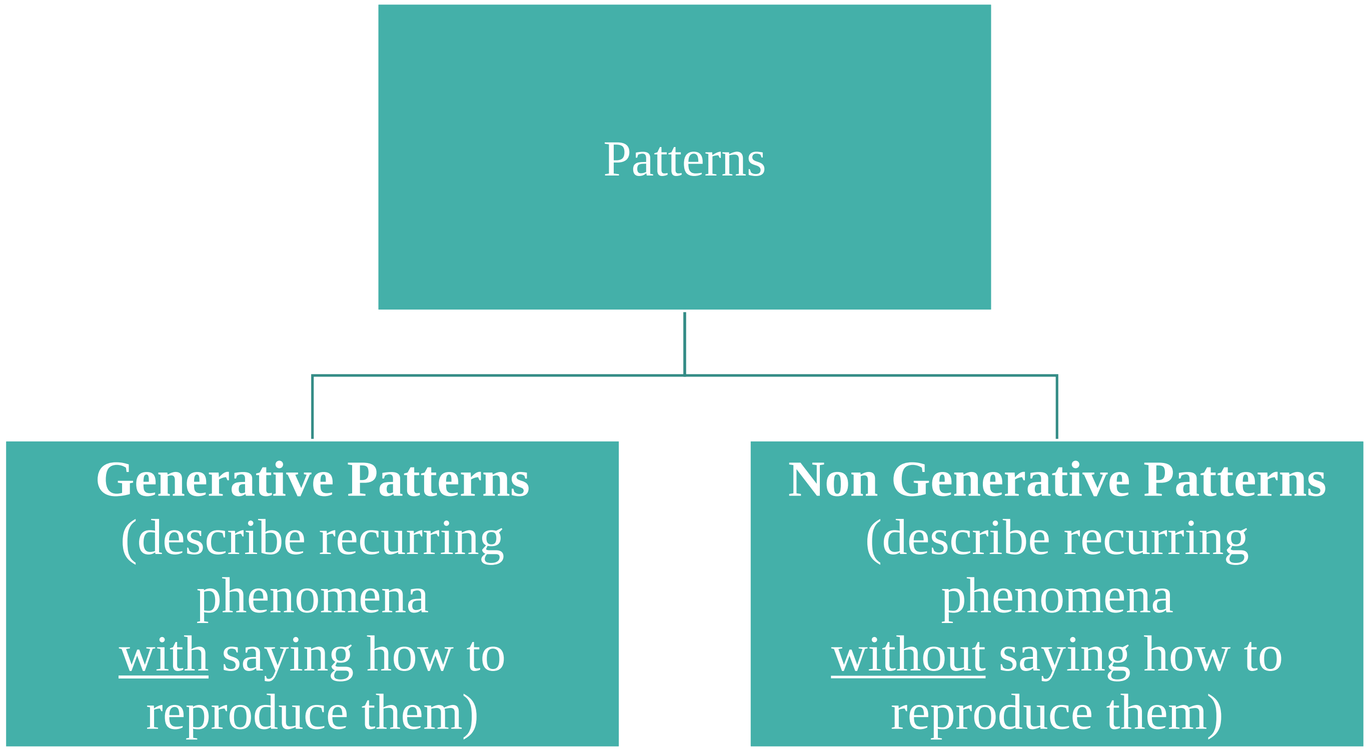
Pattern, Framework, UML

SEMESTER II TAHUN AJARAN 2022/2023

Patterns

- It is an instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system of forces.
- Good Pattern will do the following:
  - It solves a problem.
  - It is a proven concept.
  - The Solution is not obvious.
  - It describes a relationship.
  - The pattern has a significant human component.

Patterns



Patterns

Generative Patterns
(describe recurring phenomena with saying how to reproduce them)

Non Generative Patterns
(describe recurring phenomena without saying how to reproduce them)

Patterns Template

- Essential Components should be clearly recognizable on reading a pattern:
  - Name
  - Problem
  - Context
  - Forces
  - Solution
  - Examples
  - Resulting context
  - Rationale
  - Related Patterns
  - Known uses

Organizing The Pattern Catalog

Creational	Structural	Behavioral
Abstract Factory	Adapter	Chain of Responsibility
Builder	Bridge	Command
Factory Method	Composite	Interpreter
Prototype	Decorator	Iterator
Singleton	Façade	Mediator
	Flyweight	Memento
	Proxy	Observer
		State
		Strategy
		Template Method
		Visitor

E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

KNOWLEDGE & SOFTWARE ENGINEERING

Frameworks (I)

- Is a set of cooperating classes that make up a reusable design **for a specific class** of software
- The framework **dictates the architecture** of your application
  - Emphasize design reuse over code reuse
- If applications hard to design, and toolkits are harder, then **frameworks** are **hardest** of all
- A framework that using <u>design patterns</u> is far more likely to achieve high levels of design and code reuse than one that doesn’t
  - Mature framework usually **incorporate several design patterns**

Frameworks (2)

- Way of delivering application development patterns to support best practice sharing during application development.
- Can be viewed as the implementation of a system of design patterns.
- Benefits of Frameworks:
  - Reusability
  - Modularity
  - Extensibility
  - Inversion of Control

Framework vs Design Pattern

- Design patterns are more abstract than frameworks
  - Frameworks can be embodied in code, but only example of patterns can be embodied in code
- Design patterns are smaller architectural elements than frameworks
  - A typical framework contain several design patterns
- Design pattern are less specialized than frameworks
  - Framework always have a particular application domain

Unified Modeling Language (UML)

What is UML?

- UML (Unified Modeling Language)
  - Nonproprietary standard for modeling software systems, OMG
  - Convergence of notations used in object-oriented methods
    - OMT (James Rumbaugh and collegues)
    - Booch (Grady Booch)
    - OOSE (Ivar Jacobson)
- Current Version: UML 2.2
  - Information at the OMG portal http://www.uml.org/
- Commercial tools: Rational (IBM),Together (Borland), Visual Architect (business processes, BCD)
- Open Source tools: ArgoUML, StarUML, Umbrello
- Commercial and Opensource: PoseidonUML (Gentleware)

KNOWLEDGE & SOFTWARE ENGINEERING

Model

- Model is an iterative process.
- It can represent static or dynamic situations.

$$\begin{tikzcd}
 & \text{Model} \arrow[dl] \arrow[dr] & \\
\text{Static} & & \text{Dynamic}
\end{tikzcd}$$

Provides a system's parameters at rest or at a specific point in time.

(e.g.) class diagram

Represents a system's behaviors that, taken together, reflect its behavior over time.

(e.g.) interaction & activity diagrams

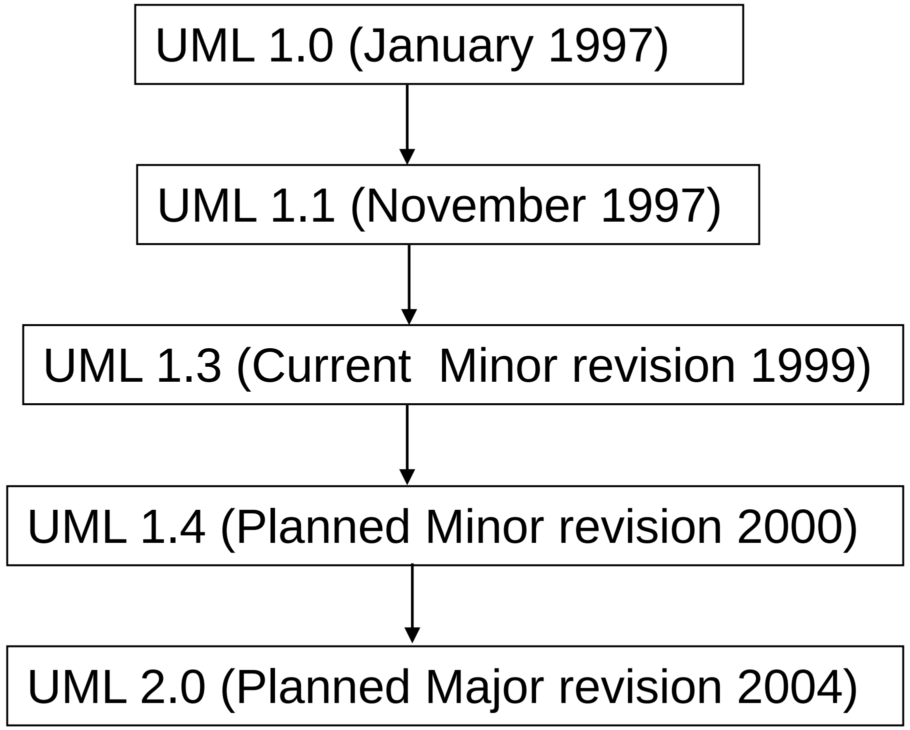
What is Unified Modeling Language (UML)?

- The UML is a graphical/standard language for
  - visualizing,
  - specifying,
  - constructing
  - documenting

  the artifacts of a software system

History of UML

- 1980 – 1990 → Many different methodologies
  1. Booch method by Grady Booch
  2. Object Modeling Technique (OMT) by Jim Rumbaugh
  3. Object Oriented Software Engineering (OOSE) by Ivar Jacobson
- Each method had its strengths & weaknesses.
  1. *Booch* was great in *design*
  2. *OMT & OOSE* were great in *analysis*

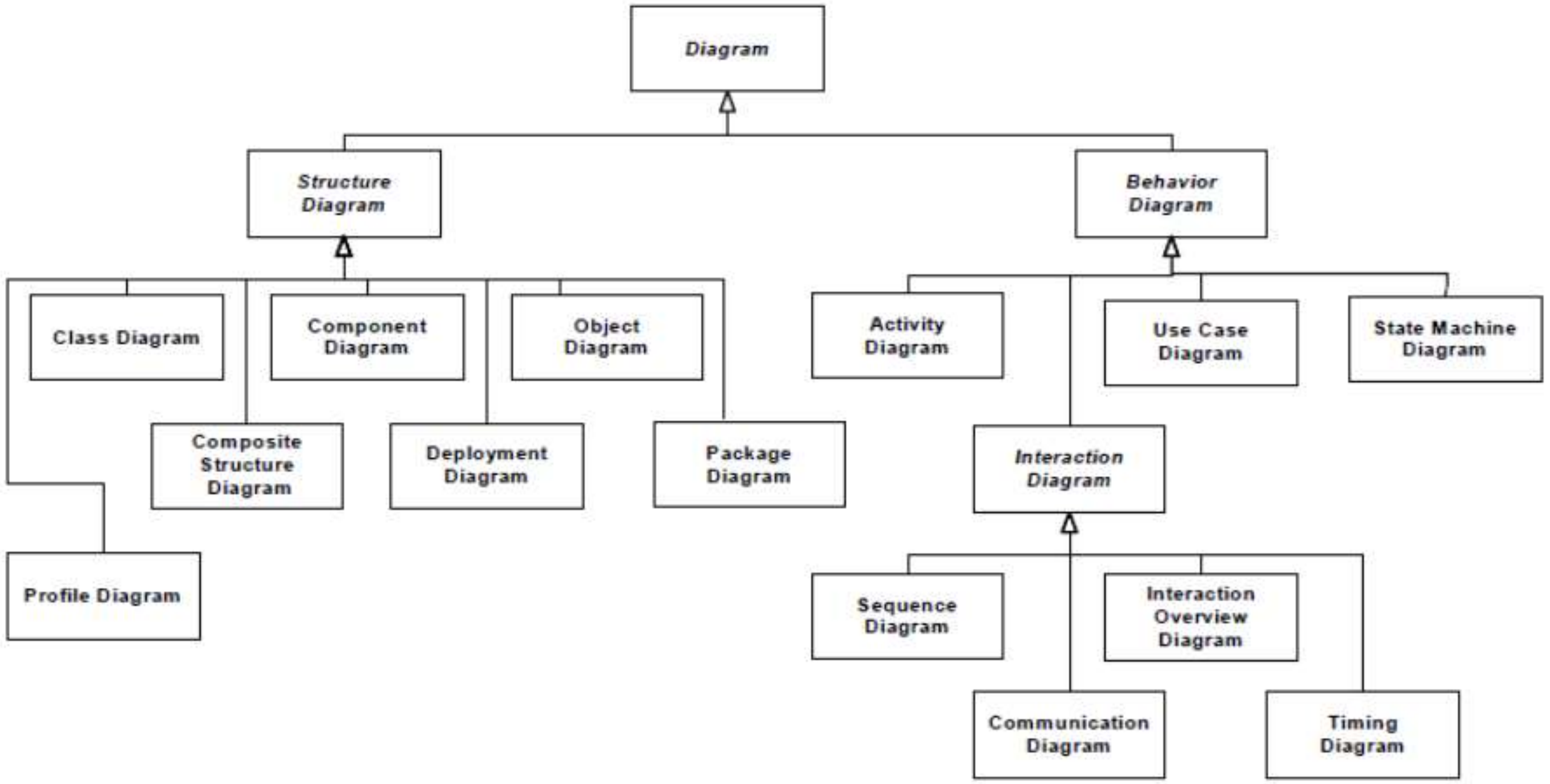

UML 1.0 (January 1997)
UML 1.1 (November 1997)
UML 1.3 (Current Minor revision 1999)
UML 1.4 (Planned Minor revision 2000)
UML 2.0 (Planned Major revision 2004)

UML 2.5

- **Structure diagrams** show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other.
  - The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.
- **Behavior diagrams** show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

KNOWLEDGE & SOFTWARE ENGINEERING

UML Diagrams Taxonomy

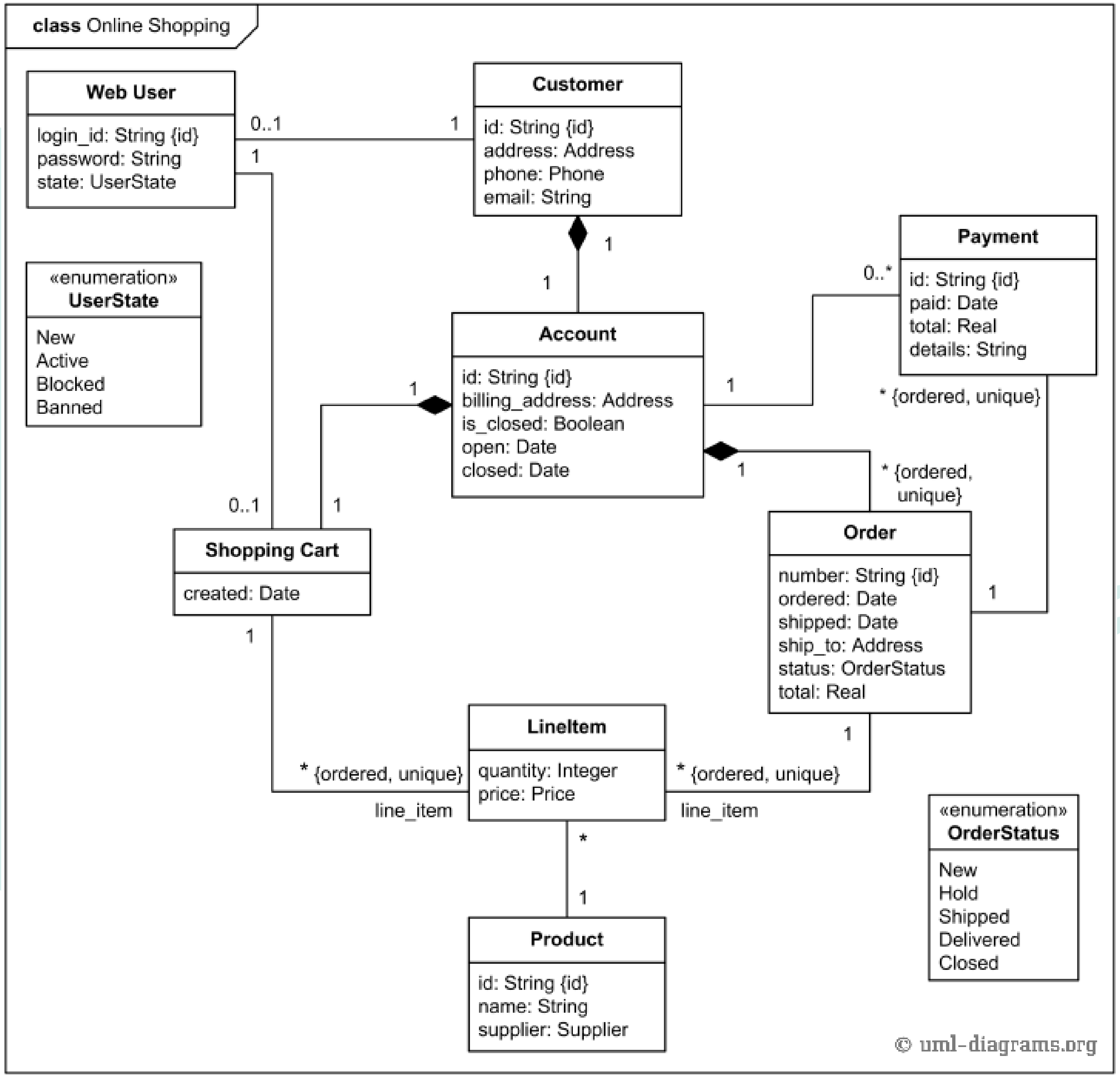


Class Diagram

- Shows structure of the designed system, subsystem or component as related classes and interfaces, with their features, constraints and relationships - associations, generalizations, dependencies, etc

Class Diagram

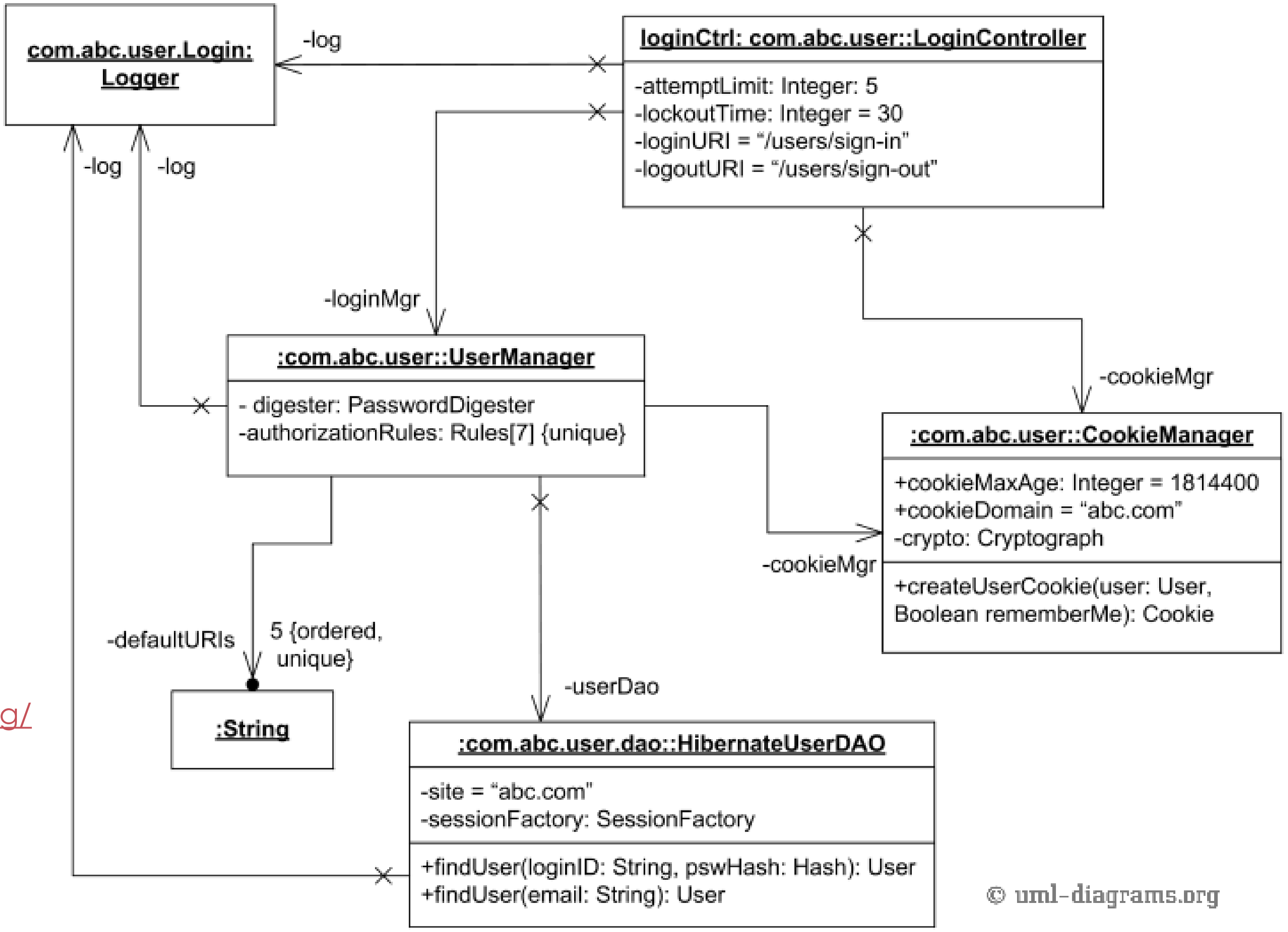
Taken from
http://www.uml-diagrams.org/



Object Diagram

- **Instance level class diagram** which shows instance specifications of classes and interfaces (objects), slots with value specifications, and links (instances of association)

Object Diagram



com.abc.user.Login: Logger
-log
loginCtrl: com.abc.user::LoginController
-attemptLimit: Integer: 5
-lockoutTime: Integer = 30
-loginURI = "/users/sign-in"
-logoutURI = "/users/sign-out"
-log
-log
-loginMgr
-cookieMgr
:com.abc.user::UserManager
- digester: PasswordDigester
-authorizationRules: Rules[7] {unique}
:com.abc.user::CookieManager
+cookieMaxAge: Integer = 1814400
+cookieDomain = "abc.com"
-crypto: Cryptograph
+createUserCookie(user: User, Boolean rememberMe): Cookie
-cookieMgr
-defaultURIs
5 {ordered, unique}
:String
-userDao
:com.abc.user.dao::HibernateUserDAO
-site = "abc.com"
-sessionFactory: SessionFactory
+findUser(loginID: String, pswHash: Hash): User
+findUser(email: String): User

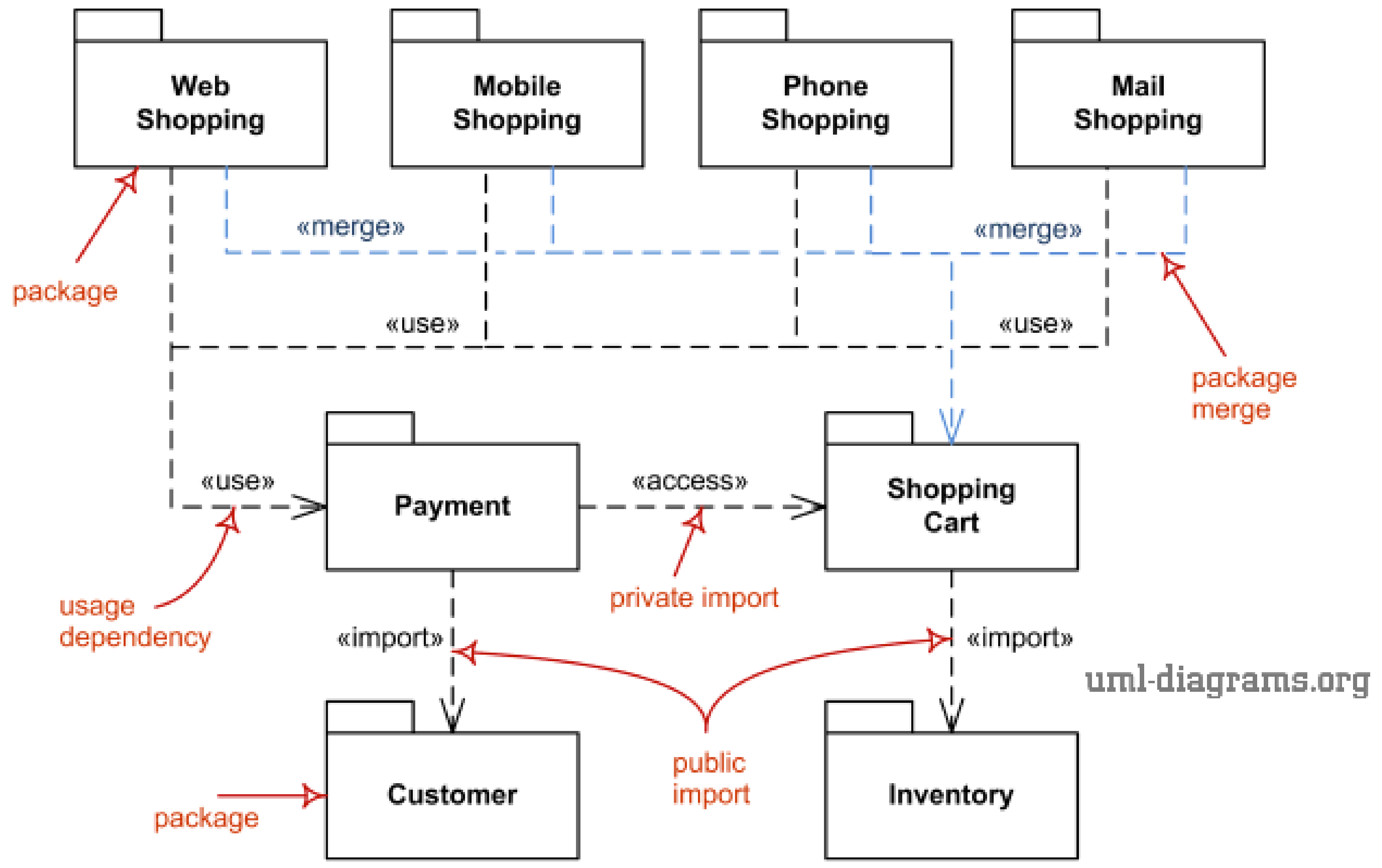
© uml-diagrams.org

Taken from
http://www.uml-diagrams.org/

Package Diagram

- Shows how model elements are **organized into packages** as well as the relationships between the packages, package, packageable element, dependency, element import, package import, package merge.

Package Diagram



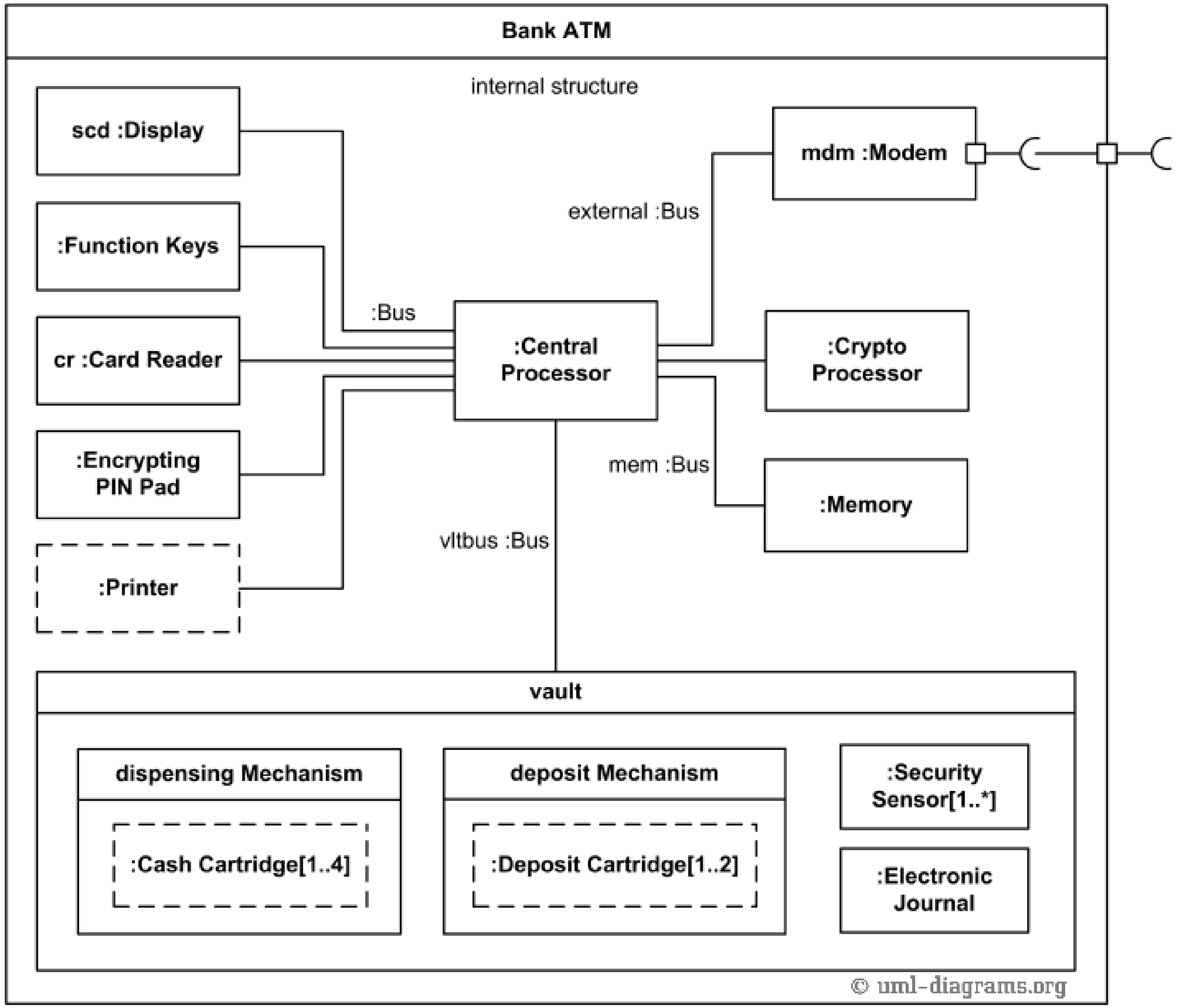
Taken from http://www.uml-diagrams.org/

Composite Structure Diagram

- Depicts
  - the **internal structure** of a classifier (such as a class, component, or use case)
  - including the **interaction points** of the classifier to other parts of the system
  - a **behavior** of a collaboration

KNOWLEDGE & SOFTWARE ENGINEERING

Composite Structure Diagram

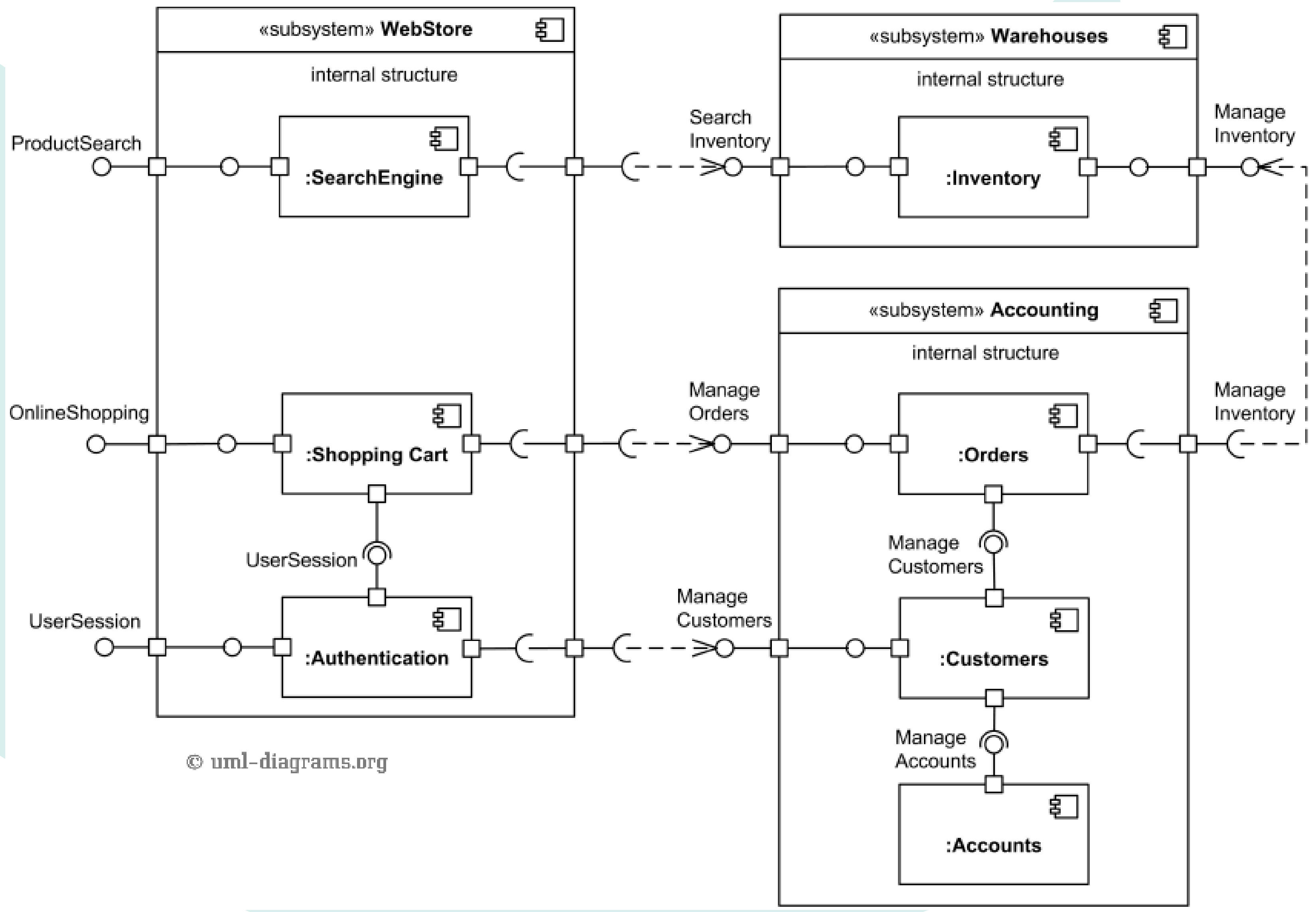


Taken from
http://www.uml-diagrams.org/

Component Diagram

- Depicts the **components that compose an application**, system, or enterprise.
- The components, their interrelationships, interactions, and their public interfaces are depicted

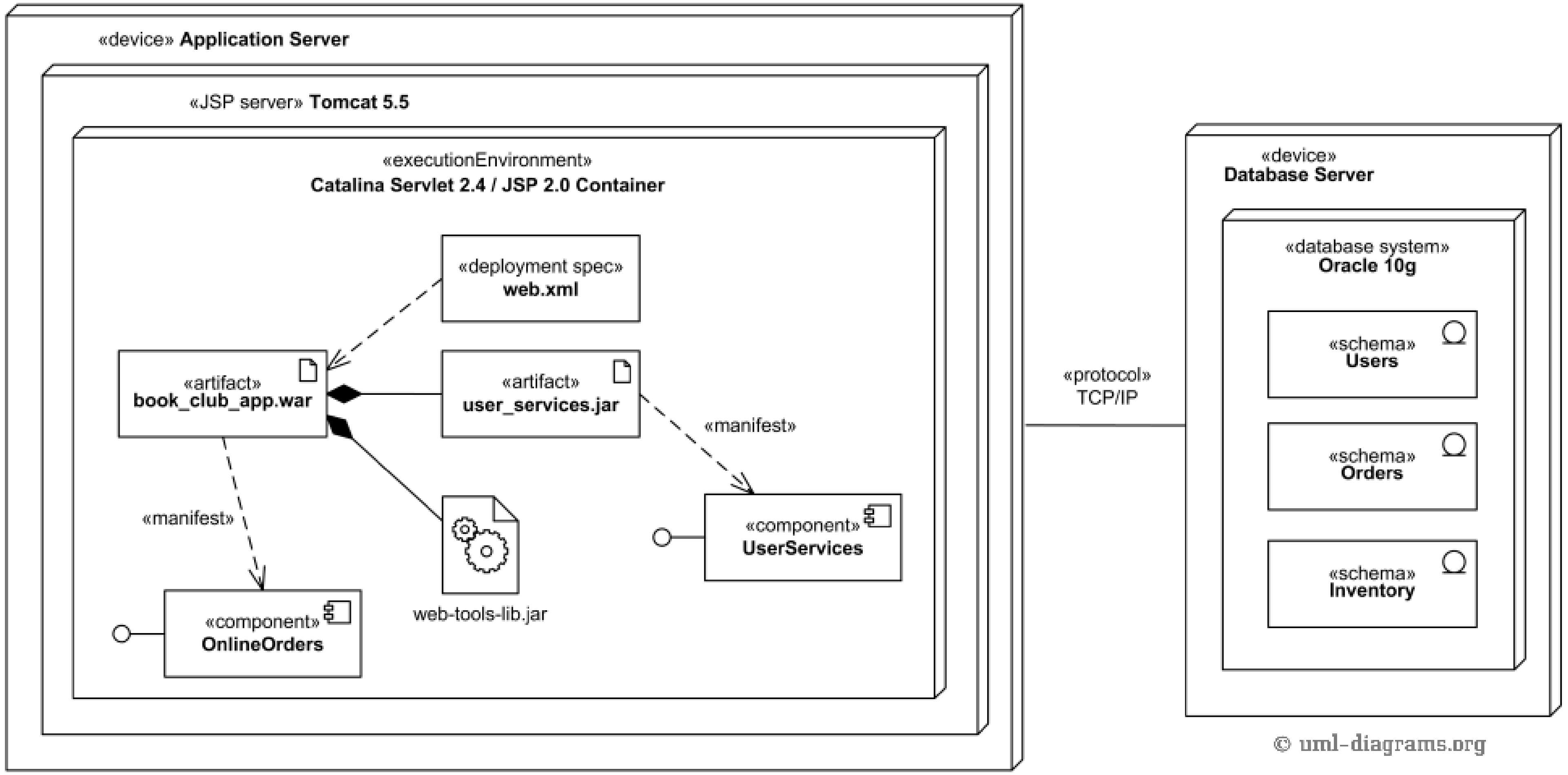
Component Diagram



Taken from
http://www.uml-diagrams.org/

Deployment Diagram

- Shows architecture of the system as **deployment (distribution) of software artifacts** to deployment targets.
- This includes **nodes**, either hardware or software execution environments, as well as the **middleware connecting them**

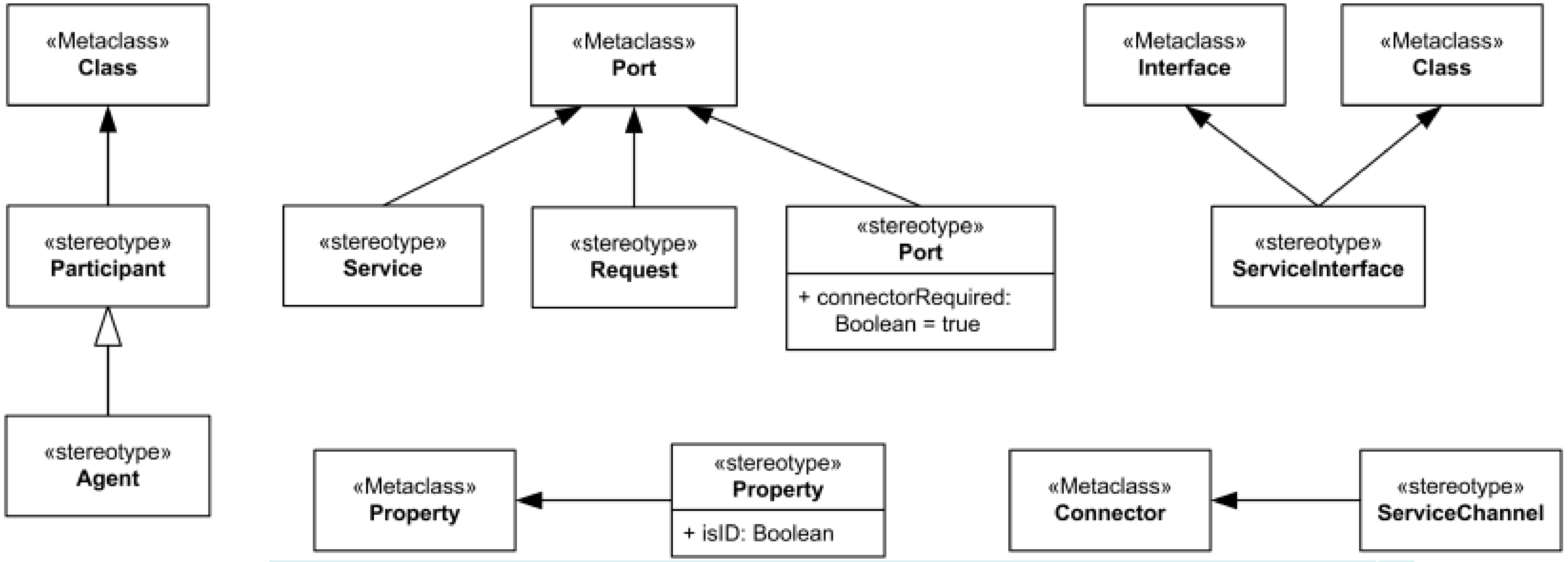


Taken from
http://www.uml-diagrams.org/

Deployment Diagram

Profile Diagram

- Auxiliary UML diagram which allows to **define custom stereotypes, tagged values, and constraints** as a lightweight **extension mechanism** to the UML standard.
- Profiles allow **to adapt the UML metamodel** for different
  - platforms (such as J2EE or .NET), or
  - domains (such as real-time or business process modeling)



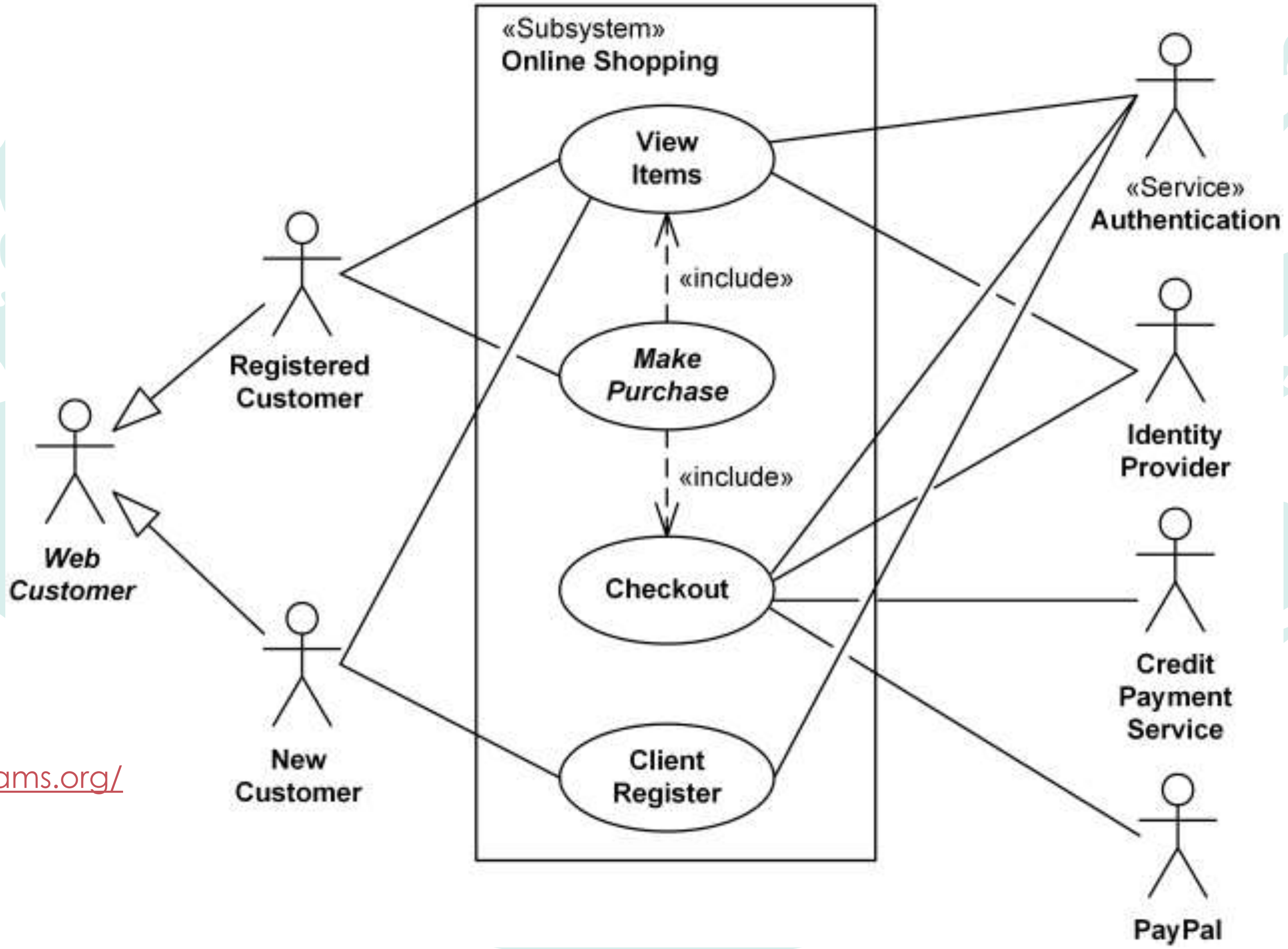
Taken from
http://www.uml-diagrams.org/

Profile Diagram

Use Case Diagram

- Describes a **set of actions** (use cases) that some system or systems (subject) should or can perform in collaboration with one or more **external users** of the system (actors) to provide some observable and valuable results to the actors or other stakeholders of the system(s)

Use case Diagram

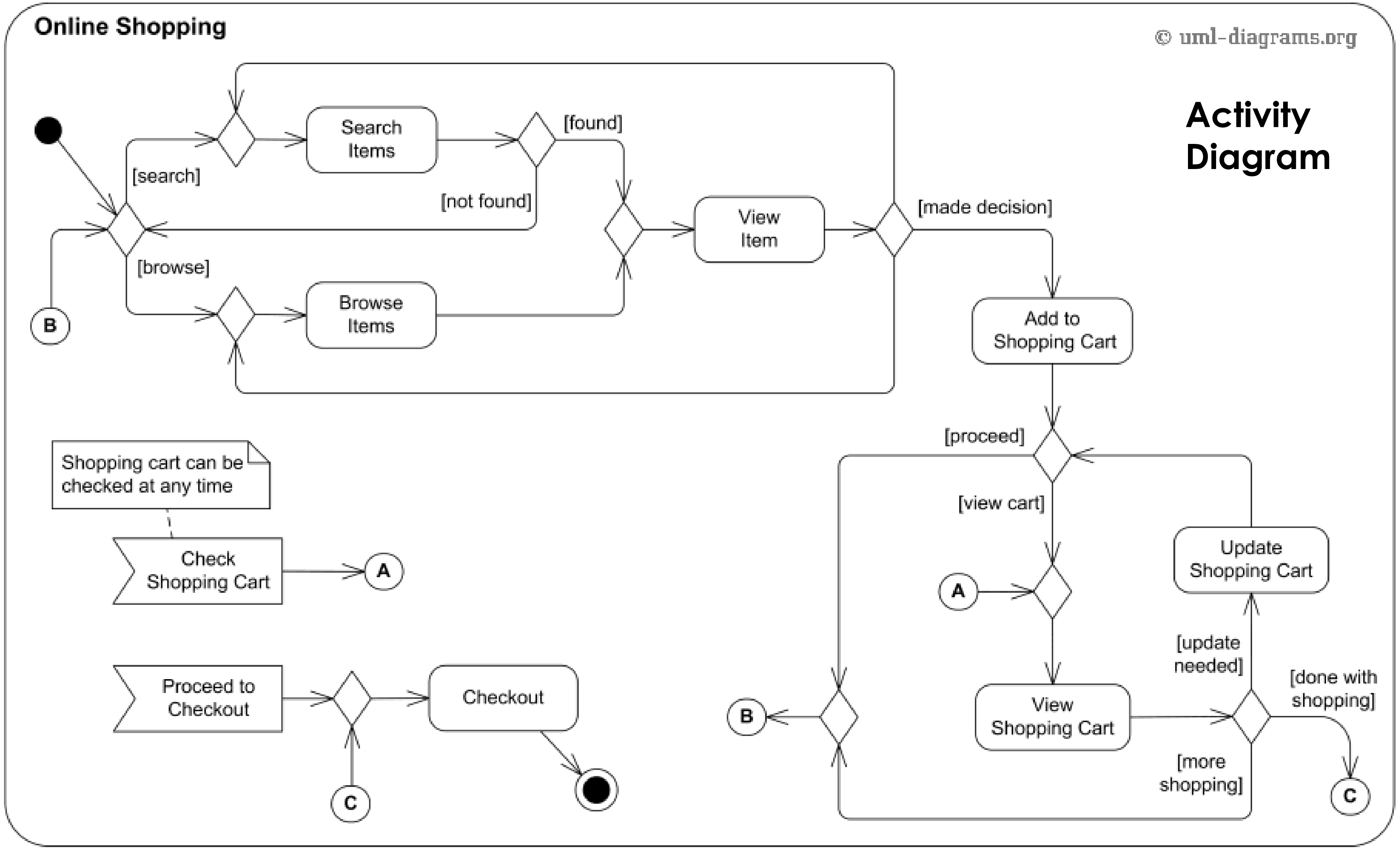


Taken from
http://www.uml-diagrams.org/

Activity Diagram

- Shows sequence and conditions for **coordinating lower-level behaviors**, rather than which classifiers own those behaviors.
- These are commonly called **control flow** and **object flow** models

Activity Diagram

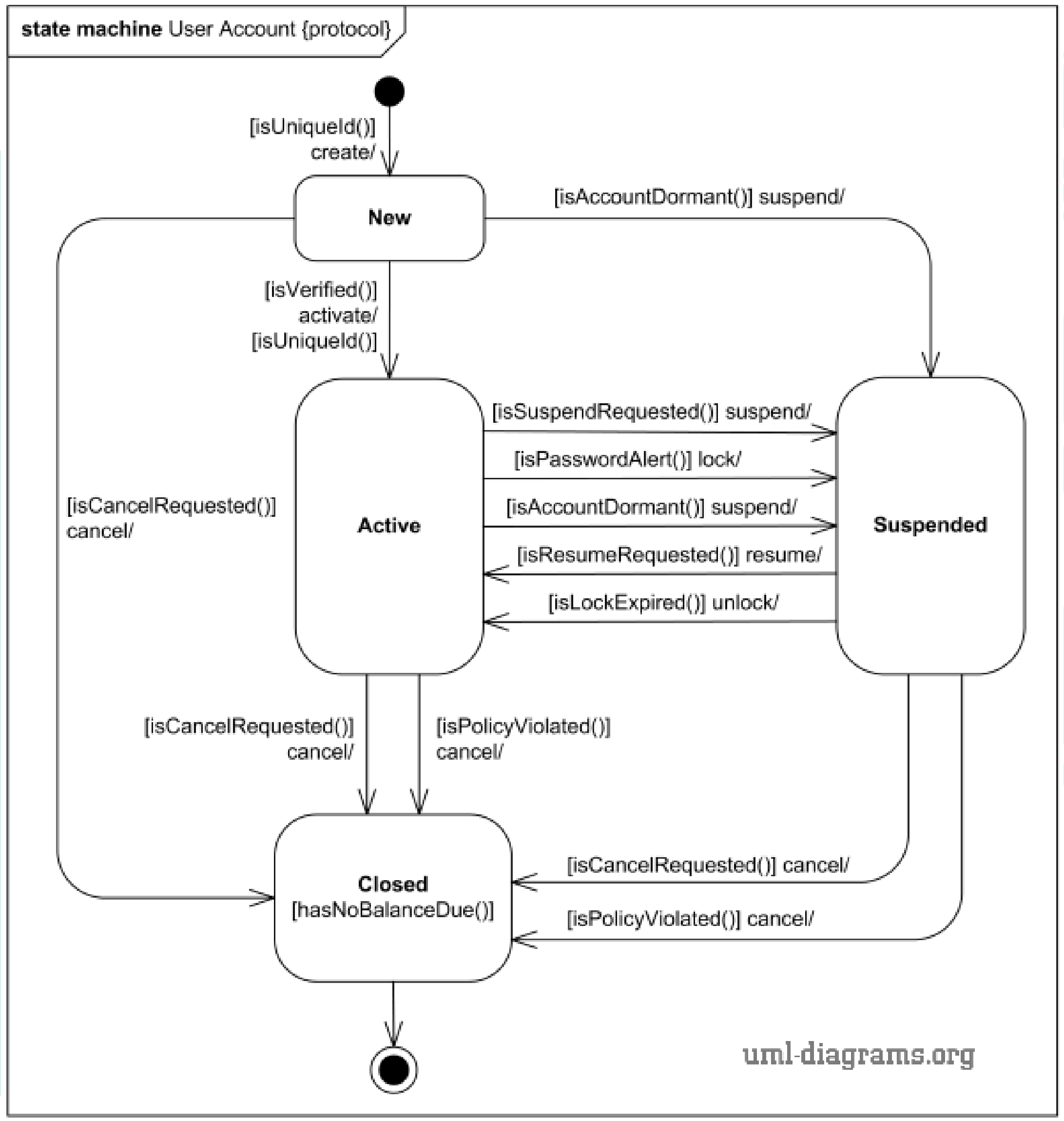


Taken from
http://www.uml-diagrams.org/

State Machine Diagram

- Describes the **states** an object or **interaction** may be in, as well as the **transitions** between states.
- Used for modeling discrete behavior/interaction through **finite state transitions**

State Machine Diagram



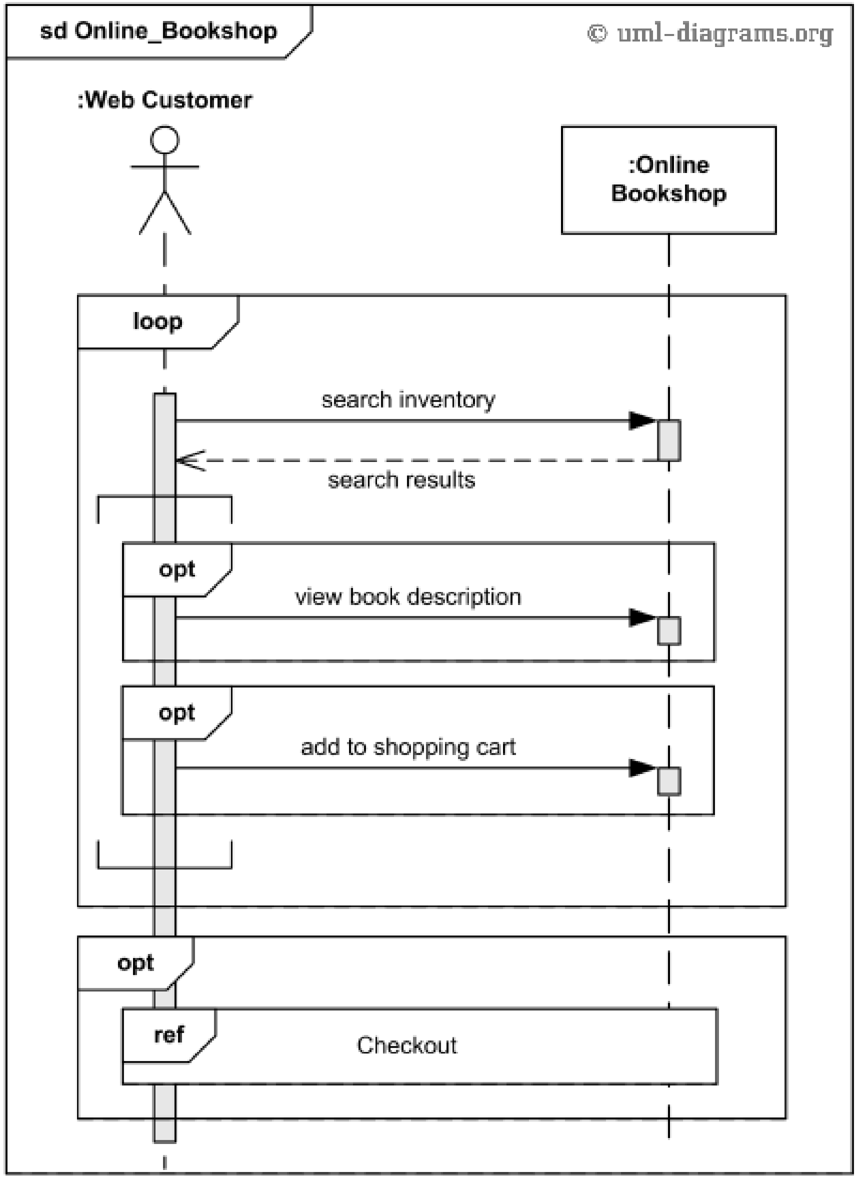
state machine User Account {protocol}
[isUniqueId()] create/
New
[isAccountDormant()] suspend/
[isVerified()] activate/ [isUniqueId()]
[isCancelRequested()] cancel/
Active
[isSuspendRequested()] suspend/
[isPasswordAlert()] lock/
[isAccountDormant()] suspend/
[isResumeRequested()] resume/
[isLockExpired()] unlock/
Suspended
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
Closed [hasNoBalanceDue()]
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
uml-diagrams.org

Taken from http://www.uml-diagrams.org/

Sequence Diagram

- Most common kind of interaction diagrams which focuses on the **message interchange between lifelines** (objects).

Sequence Diagram

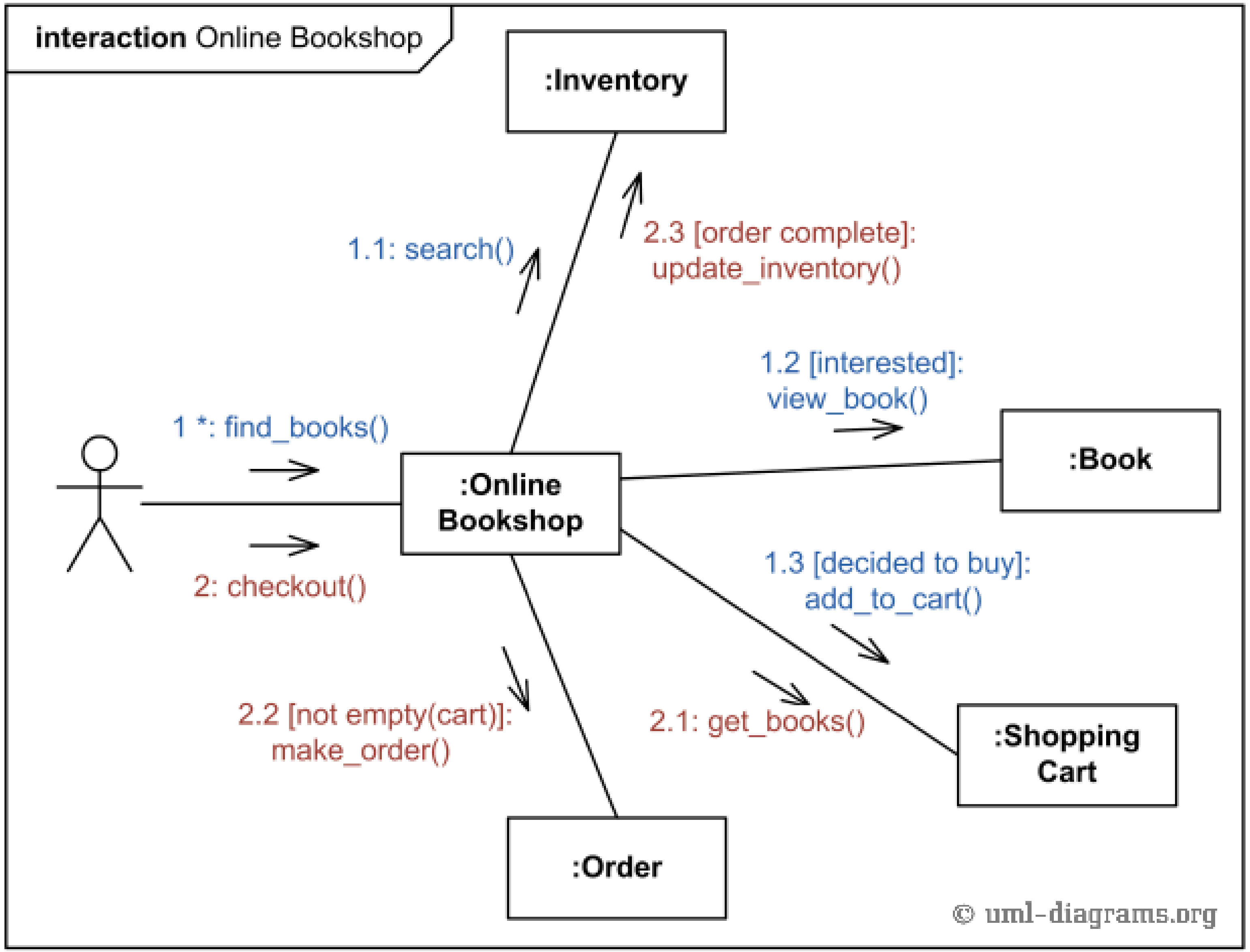


Taken from
http://www.uml-diagrams.org/

Communication Diagram

- Focuses on the **interaction between lifelines** where the architecture of the internal structure and how this corresponds with **the message passing** is central.
- The **sequencing of messages** is given through a sequence **numbering** scheme.

Communication Diagram


interaction Online Bookshop
:Inventory
1.1: search()
2.3 [order complete]: update_inventory()
1.2 [interested]: view_book()
:Book
1 *: find_books()
:Online Bookshop
2: checkout()
1.3 [decided to buy]: add_to_cart()
2.2 [not empty(cart)]: make_order()
2.1: get_books()
:Shopping Cart
:Order
© uml-diagrams.org

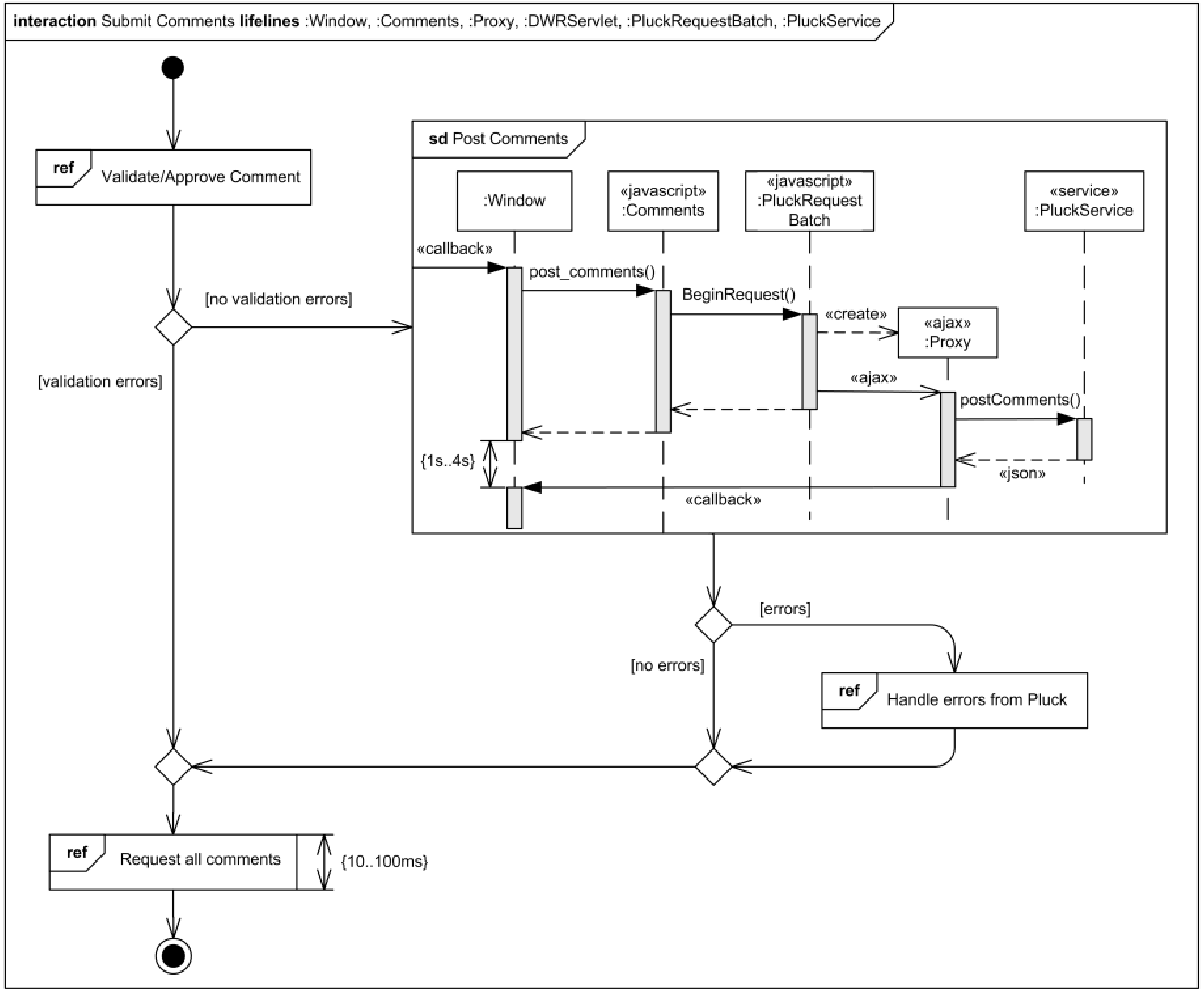
Taken from http://www.uml-diagrams.org/

KNOWLEDGE & SOFTWARE ENGINEERING

Interaction Overview Diagram

- A **variant of an activity diagram** which overviews the control flow within a system or business process.
- Each node/activity within the diagram can represent another interaction diagram.
- It focuses on the **overview** of the flow of control where the nodes are interactions or interaction uses.
- The lifelines and the messages do not appear at this overview level

Interaction Overview Diagram

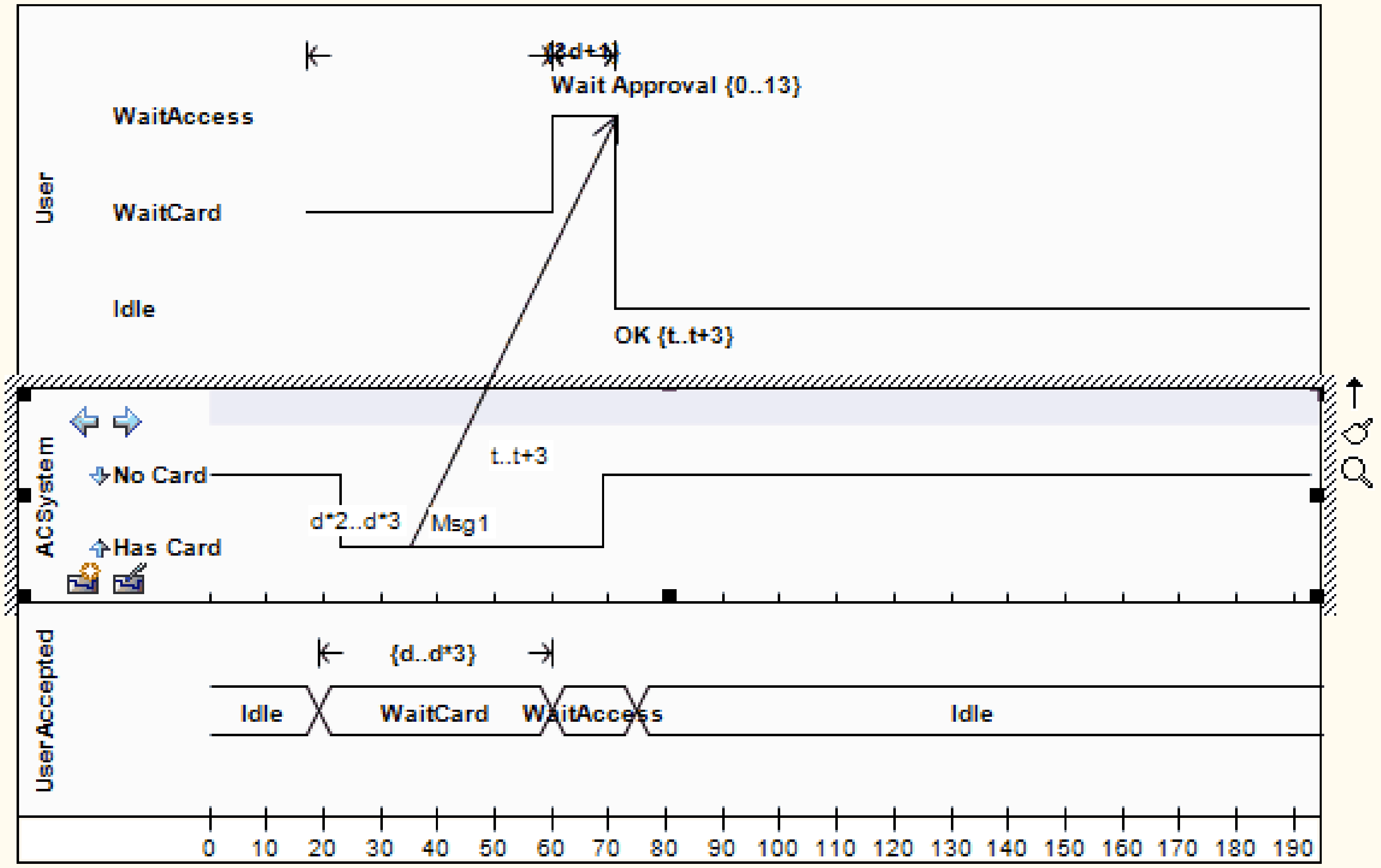


Taken from http://www.uml-diagrams.org/

Timing Diagram

- Depicts the **change in state or condition** of a classifier instance or role **over time**.
- Typically used to show the change in state of an object over time **in response to external events**.
- It focuses on conditions changing within and among lifelines along a **linear time axis**

Timing Diagram



Taken from
http://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/timingmessage.html