

ВЪВЕДЕНИЕ В UML

Unified Modelling Language
(Унифициран език за моделиране)

СУ „Св. Климент Охридски“
Факултет по Математика и Информатика
Увод в Софтуерното Инженерство

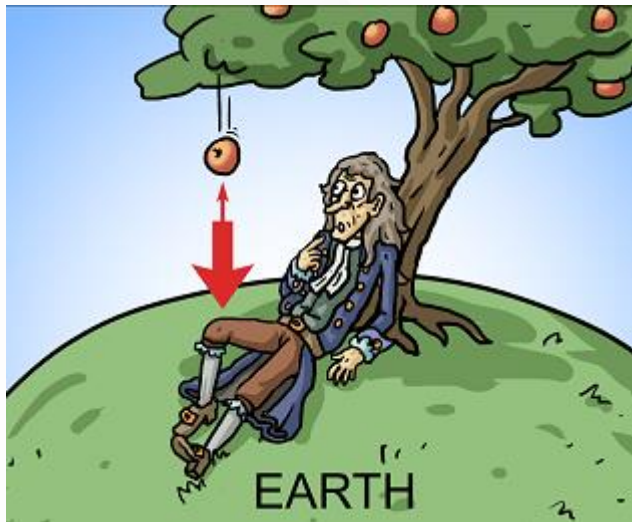
Преговор

- Софтуерна архитектура
 - Съвкупност от различни изгледи на системата, които се състоят от софтуерни елементи, външно видимите характеристики на тези елементи, и връзките, които съществуват между тях
- А какво означава изглед?

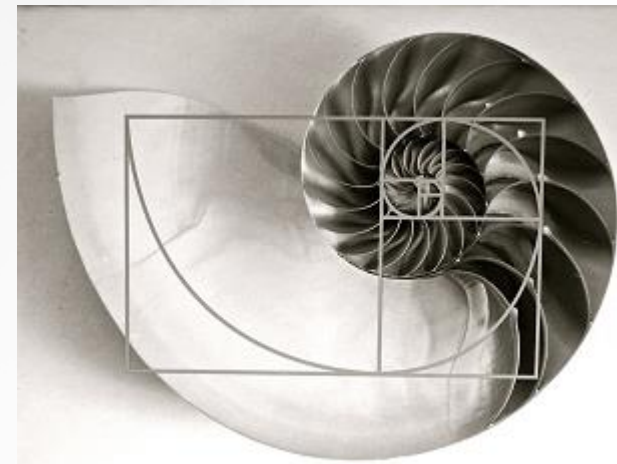
Защо е нужно да се документира архитектурата

- Комуникация между заетите в проекта по разработка
 - Анализ на системата
 - Многократна употреба
-
- А как се документира архитектурата?

Понятие за модел



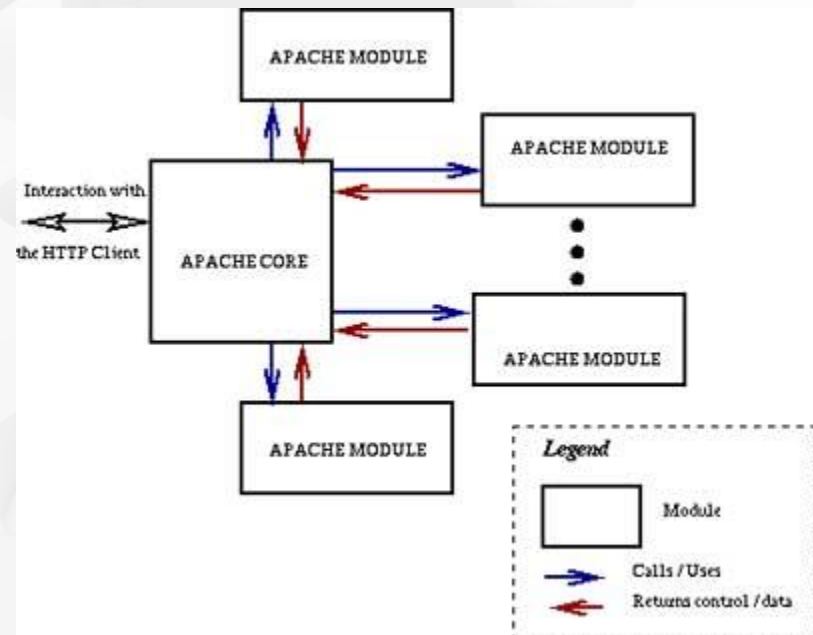
$$\frac{\partial}{\partial t} T(x, y, t) = \chi \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) T(x, y, t) + S(x, y, t)$$



А какви са моделите в софтуерните системи?

```
if (S < 0 || S == E || S > E)
    return;

int temp = 0;
for (int i = E; i >= S; i--) {
    temp = arr[S];
    arr[S] = arr[i];
    arr[i] = temp;
    S++;
}
return;
```



Моделиране

- Моделиране : процес на описание на системата чрез нейния модел (концептуален, математически или базиран на имитация) и симулация на системни дейности чрез прилагане на модела върху набор от данни .
- Моделите представят реални системи и процеси, които трудно да се наблюдават директно. Моделирането е средство за разбирането им.
- В процеса на моделиране описваме моделирания обект само с характеристиките, които са пряко свързани с това , което ни интересува .

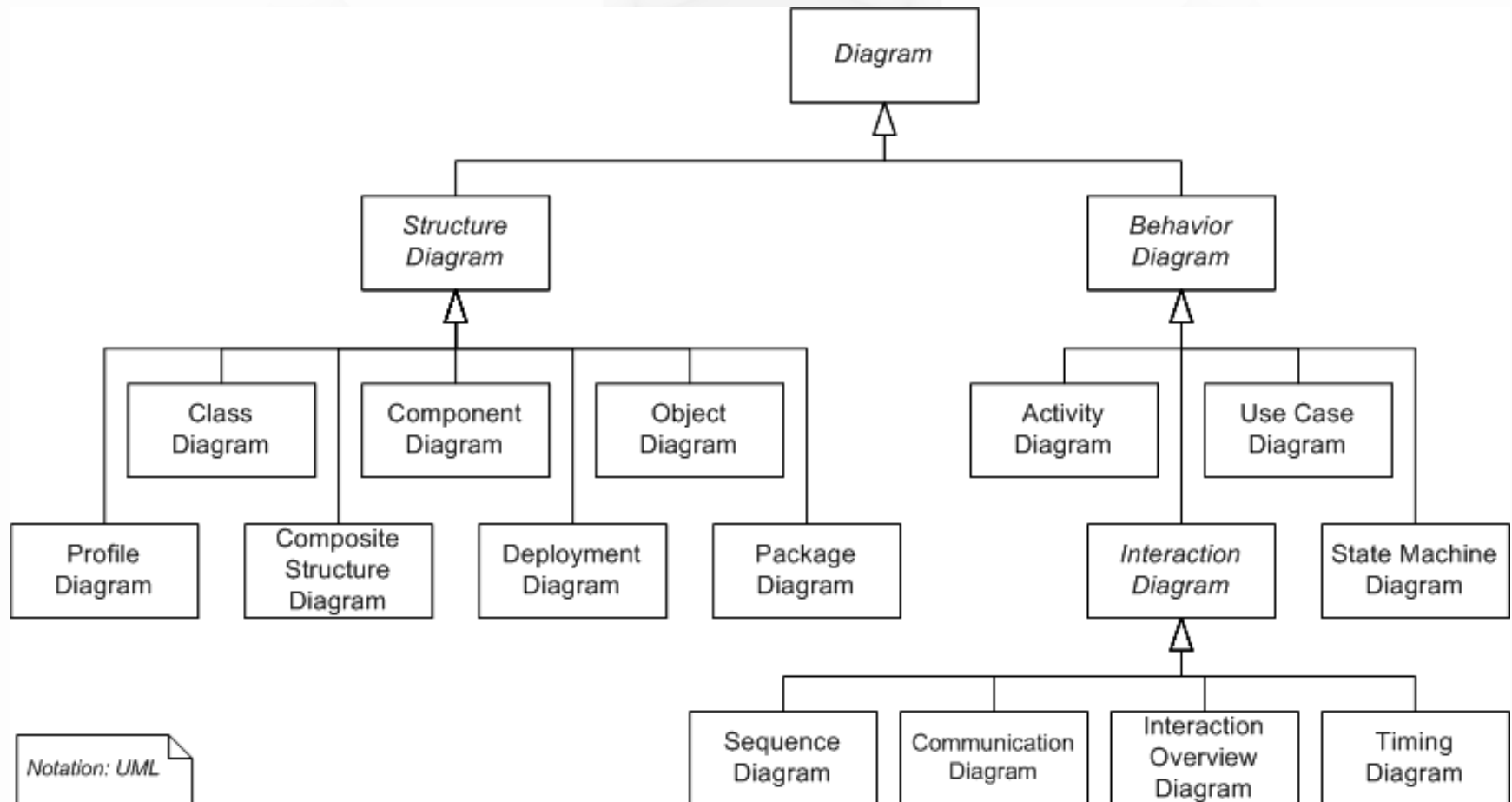
Унифициран език за моделиране

- Няколко нотации за описание на обектни системи се появяват в края на миналия век
- UML ги обединява
- Предоставя средства за описание на различни модели, които се създават по време на обектно-ориентирания анализ и дизайн
- Де-факто стандарт в ОО разработване

Диаграми в UML

- UML е свързан с обектно-ориентиран дизайн и анализ.
- UML използва елементите и формира асоциации между тях, за да формира определени диаграми.
- Диаграмите в UML могат да бъдат класифицирани като:
 - **Структурни диаграми** – представят статични аспекти или структурата на система. Структурните диаграми включват: Компонентни диаграми, Обектни диаграми, Класови диаграми и Диаграми за внедряване.
 - **Диаграми на поведението** – представят динамични аспекти или поведение на системата. Диаграмите на поведението включват: Диаграми на случаите на употреба, Диаграми на състоянието, Диаграми на дейностите и Диаграми на взаимодействието.

Диаграмми в UML



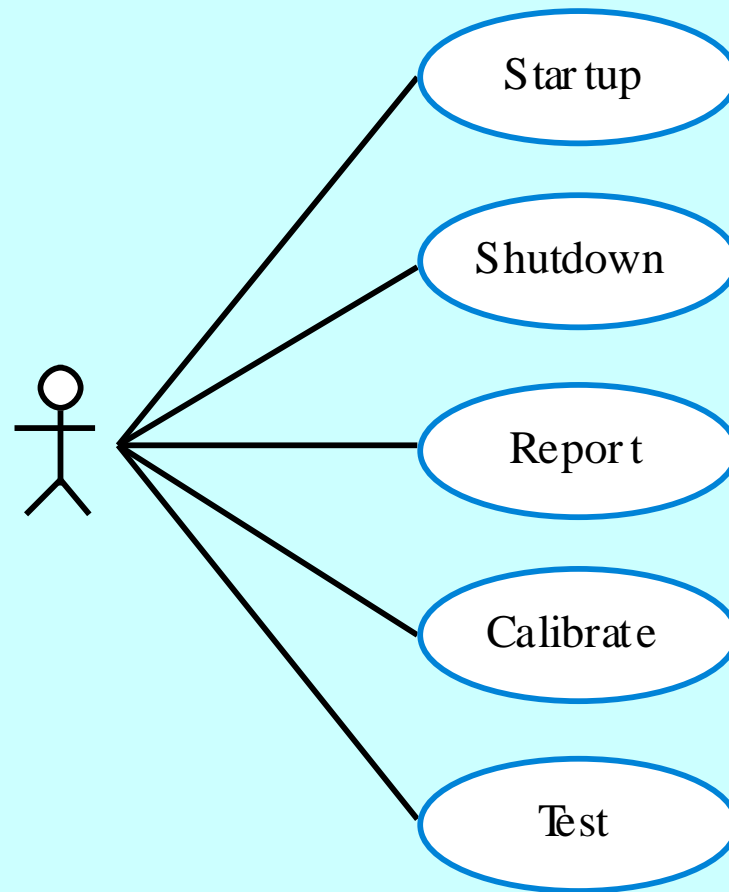
Диаграми в UML

- Use case диаграми
 - Показват взаимодействието на системата с околната ѝ среда (Environment/Context)
 - Използват се за спецификация на изискванията
- Class диаграми
 - Показват обектните класове в системата и връзките между тях
- Activity диаграми
 - Показват конкретни дейности от процесите, които протичат в системата
- Sequence диаграми
 - Показват взаимодействието между околната среда (actors) и системата или между компонентите на системата
- State diagrams
 - Показват как системата реагира на вътрешни и външни събития

Use cases

- На български – *варианти на употреба*
- Състоят се от две части
 - Диаграма на вариантите на употреба
 - Описание на вариантите на употреба – детайлно описание на всяка функция; може да бъде на естествен език или да се представи с друга UML диаграма

Примери за диаграма на вариантите на употреба

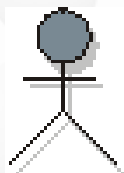


Цели

- Да се представят понятията актьор и потребителски случай
- Да се разгледа техниката на потребителски случаи при дефиниране на системните изисквания

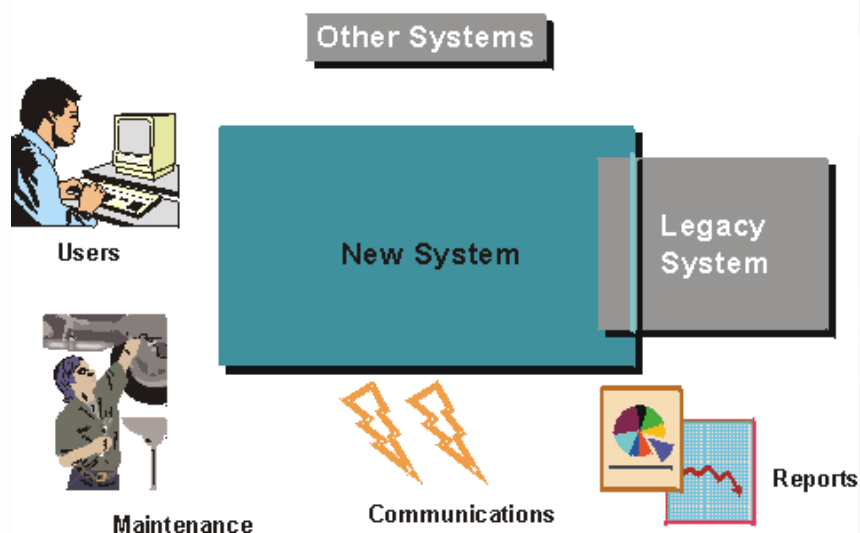
Определяне на актьорите

- Актьор



- Инстанция на актьор е някой или нещо, външно за системата, което си взаимодейства пряко със системата.
- Класът “Актьор” дефинира набор от актьори инстанции, в който всяка инстанция играе една и съща роля във взаимодействието си със системата
- За да се разбере напълно целта на системата, трябва да се знае за кого е предназначена тя или кой ще я използва. Различните типове потребители се представят като актьори.
- Актьорът е всяко нещо, което обменя данни със системата. Актьорът може да бъде потребител, външен хардуер или друга система.

Как да идентифицираме актьорите?



- Кой ще предоставя/използва/премахва информация?
- Кой ще използва тази функционалност?
- Кой е заинтересован от всяко отделно изискване?
- Каде в организацията се използва системата?
- Кой ще поддържа системата?
- Какви са външните източници на системата
- Какви други системи ще си взаимодействат с разработваната?

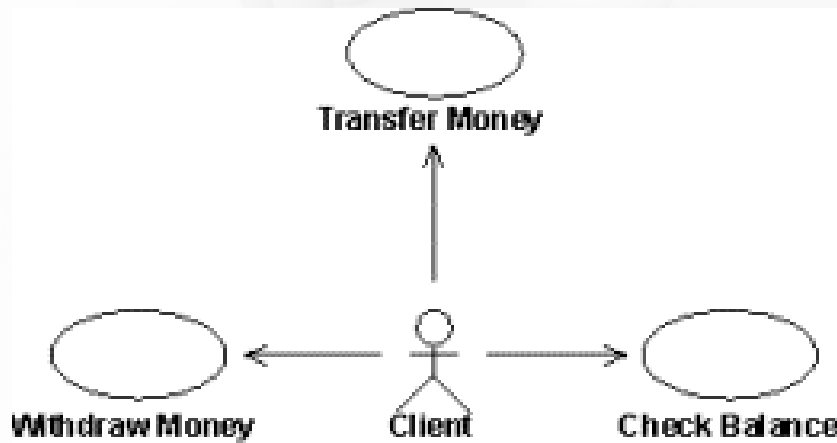
Документиране на характеристиките на актьора

- Кратко описание:
 - Какво или кого представлява актьорът
 - Защо е необходим актьорът
 - Какъв е интересът на актьора спрямо системата
- Характеристиките на актьора могат да повлияят на разработването на системата
 - Обхвата на отговорност на актьора
 - Физическата среда, в която актьорът ще използва системата
 - Броят на потребителите представяни от актьора

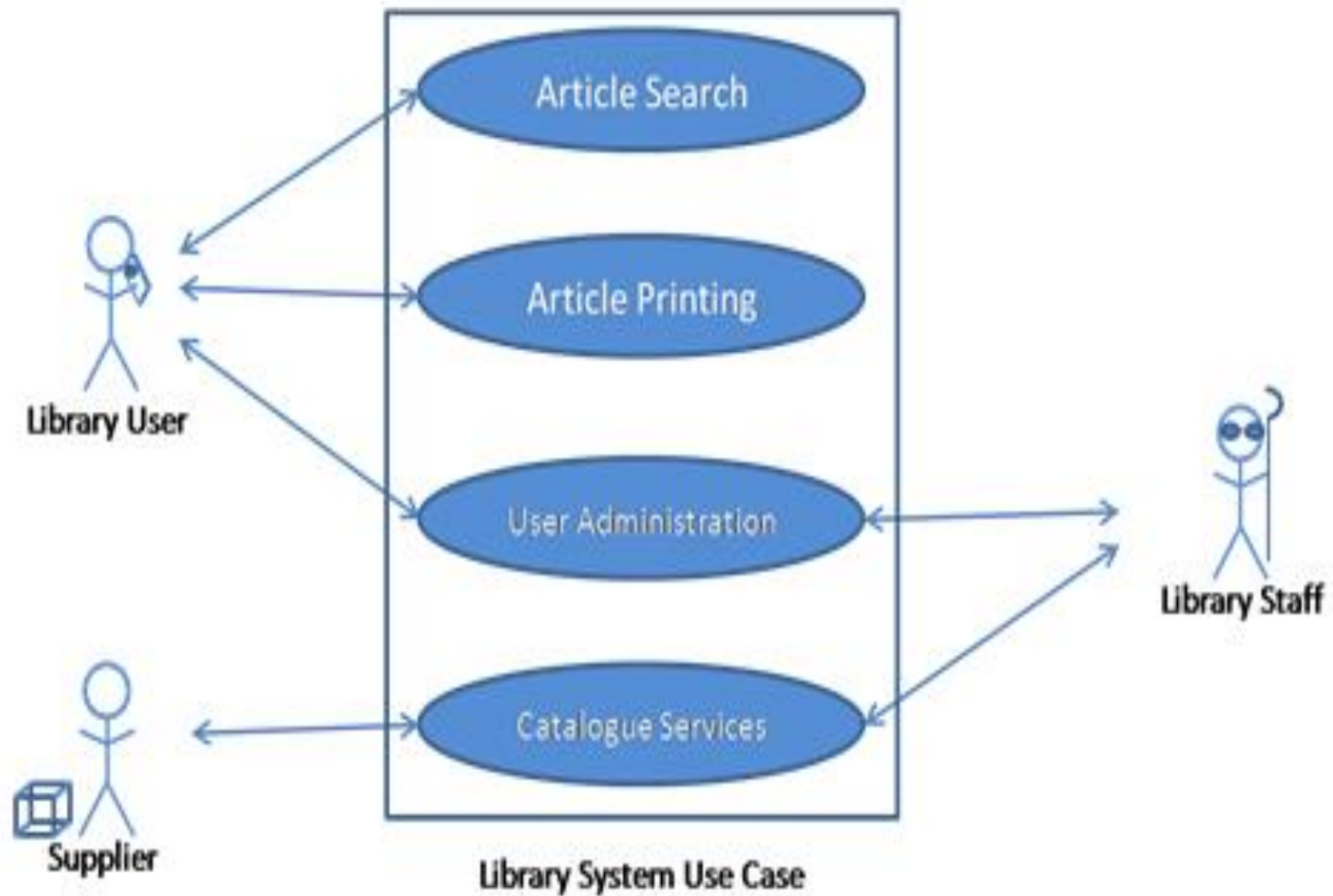
Дефиниране на потребителските случаи



- Инстанцията на потребителски случай е последователност от действия, които системата извършва, които водят до видим резултат със значение за даден актьор. Потребителският случай дефинира набор от инстанции.



Банкомат – системната функционалност е дефинирана от различни потребителски случаи, всеки от които определя специфична последователност от действия, определя какво се случва в системата, когато се изпълнява даден потребителски случай.



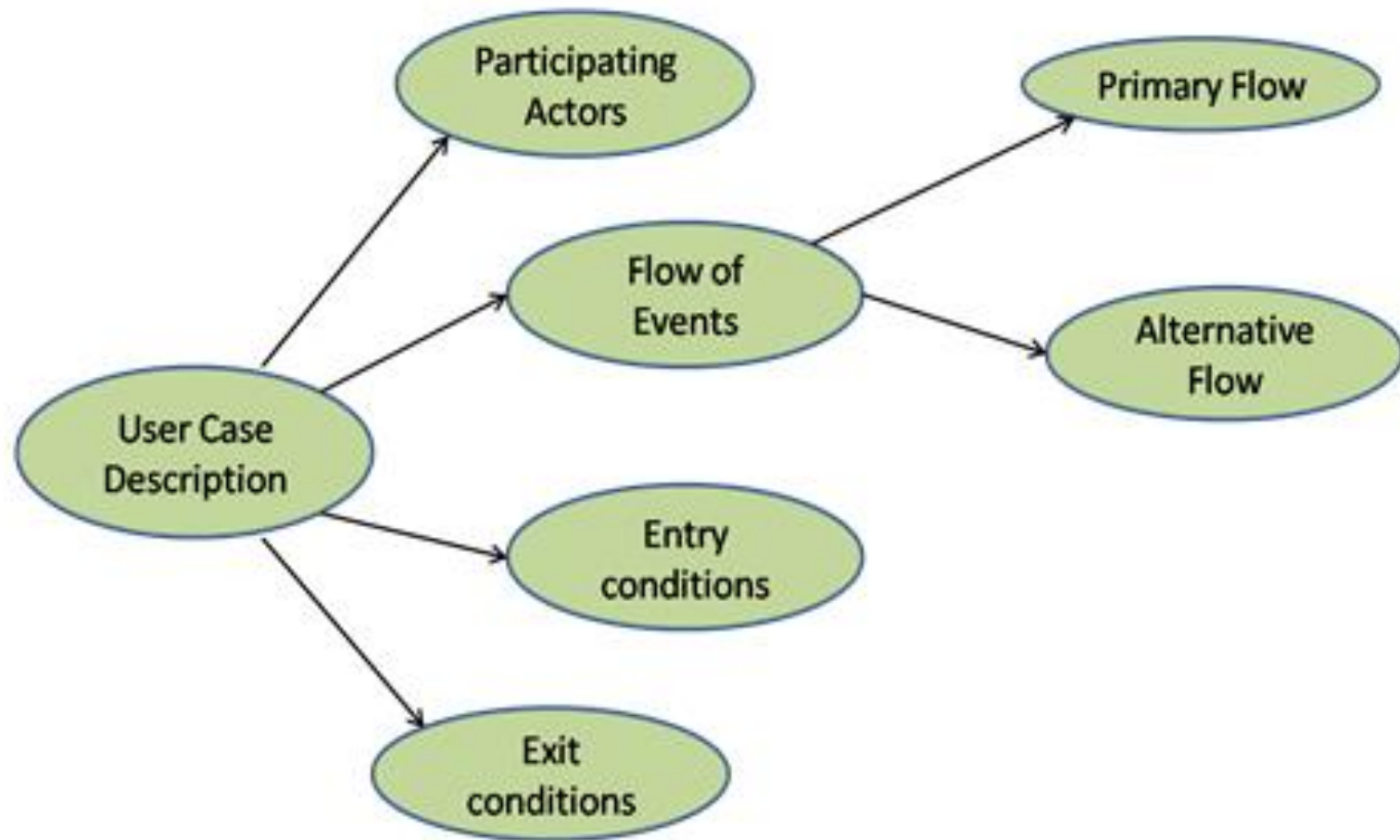
Как да намерим потребителските случаи?

- Какви са задачите на системата за всеки идентифициран актьор?
 - Трябва ли актьорът да бъде информиран за дадено събитие в системата?
 - Трябва ли актьорът да информира системата за внезапни външни промени?
 - Предоставя ли системата коректна функционалност на бизнеса?
 - Могат ли всички функции да бъдат изпълнени от идентифицираните потребителски случаи.
 - Кой потребителски случаи отговарят за поддръжката на системата?
 - Каква информация трябва да бъде модифицирана или създадена в системата?
- Видове потребителски случаи:
 - Старт и стоп на системата.
 - Поддръжка на системата.
 - Поддържане на данните съхранявани в системата.
 - Необходимата функционалност за да се модифицира поведението на системата.

Документиране на потребителските случаи – последователност от събития

- Последователността от събития в потребителския случай съдържа най-важната информация извлечена при процеса на моделиране. Тя е:
 - Описание на началото и края на потребителският случай
 - Описание на данните, които се обменят между актьора и потребителския случай
 - Описание на последователността от действия предизвикана от актьор.
 - Описание само на събитията, които принадлежат на потребителския случай.
 - Избягване на неточна терминология като “например”, “т.н.”
 - Детайлизиране на потока на събитията. На всички въпроси “какво?” трябва да има отговор в описателен стил.

Описание на вариантите на употреба



Описание на вариантите на употреба

Use Case Name	Searching for an Article
Participating Actors	Library User
Flow of Events	<ol style="list-style-type: none">1.User enters search terms2.System searches' articles from file3.System displays all articles matching terms4.User searches for article from given list and clicks on the one they need5.System displays that article to the user
Alternative Flows	<ol style="list-style-type: none">2.a System has no articles to search2.b System informs user and exits3.a System finds no articles matching users terms3.b System informs user and exits4.a User can't find article in the list4.b User exits and tries more search terms
Entry Conditions	User has correct user authentication and a valid search term
Exit Conditions	User finds correct article, or user exits searching system due to article not being found

Клас диаграми в UML

- Дефиниция
 - Клас-диаграмата описва основните типове обекти в системата и различните видове статични връзки между тях.
- Основни компоненти
 - Класове
 - Атрибути
 - Операции
 - Асоциации
 - Наследяване (Generalization)
 - Агрегация (Aggregation)
 - Композиция (Composition)

Класове

Име
Атрибути, които съдържат данни
Методи, които извършват операции върху данните

Идентификация

Състояние

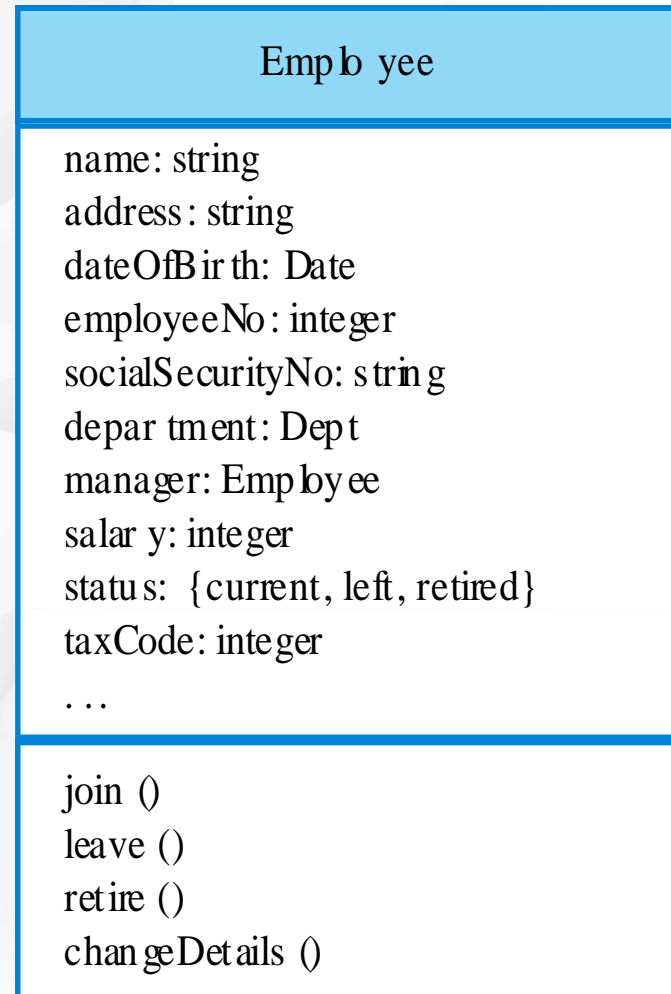
Поведение

Клас-диаграми

- Клас диаграмите се използват, най-вече тогава, когато се създава обектно-ориентиран модел на системата, който показва нейните обектни класове и връзките (асоциациите) помежду им.
- Обектният клас може да се разглежда като обобщаваща дефиниция на някакъв тип обекти в системата
 - Конкретните обекти се наричат екземпляри (инстанции – от англ. instance) на класа
 - Примери:
 - Студент, преподавател, учебна програма (в система за електронно обучение)
 - Пациент, доктор, направление, рецепта (в болнична информационна система) и т.н.

Примерна дефиниция на клас в UML

- Класовете, създавани по време на Обектно-Ориентирания (ОО) анализ, обикновено имат непълни дефиниции и представят бизнес обекти и подсистеми в реалната система
- Тези класове на анализа се използват в ОО дизайн, където обикновено се представят с множество класове на дизайна, използващи конкретна платформа за имплементацията на системата



Видимост на атрибутите

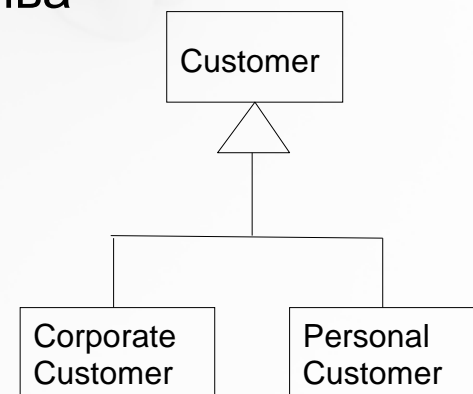
Видимостта на атрибутите може да бъде:

visibility ::= { + | - | # | ~ }

- **Public:** the attribute is visible both inside and outside the package containing the class.
- **Private:** the attribute is only visible to the class itself and to **friends** of the class
- **Protected:** the attribute is visible only to the class itself, to its subclasses, or to **friends** of the class (language dependent)
- **Package:** only classes within the same package as the container can see and use the classes.

Асоциации (associations) в клас-диаграмите

- Обектите и техните класове имат взаимоотношения с други обекти и класове
- В UML това се отбелязва чрез т.нар. асоциации
 - Асоциациите може да се доуточнят с т.нар. анотации (annotation)
 - Изобразяват се най-общо с черта
 - Честа практика е ако един клас има повече асоциирани с него класове, асоциативната черта да се обединява

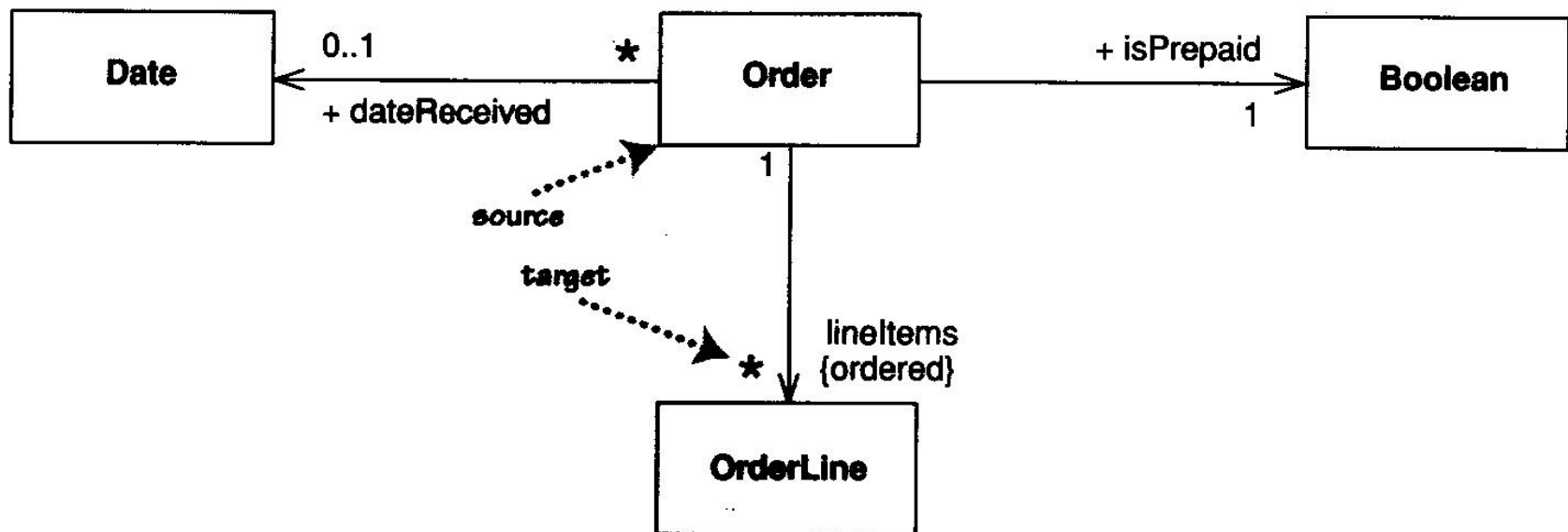
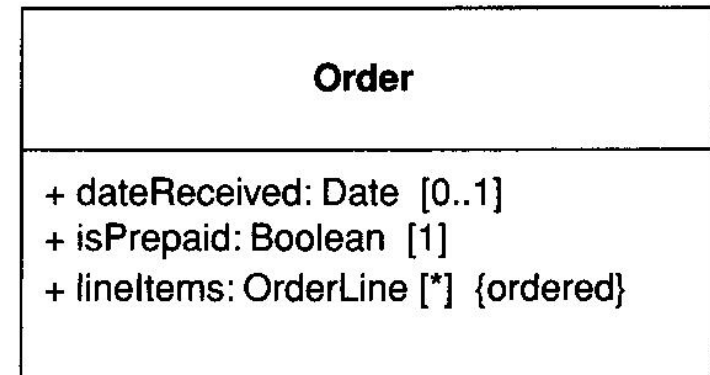


Асоциации

- Представят взаимоотношения както между класове, така и между обекти
 - Например клас person се отнася разботещите в дадена фирма (представена чрез клас company)
 - Фирмата (company) има известен брой офиси/клонове (offices).
- Представят роли
 - С текст в края на асоциативната черта
- Множественост
 - Показват колко обекти може да участват в дадено взаимодействие
 - 0, 1, *, 0..1, 1..*, 5..6, и т.н.
 - * - неограничен брой
 - Реализират се чрез масиви, списъци и т.н.

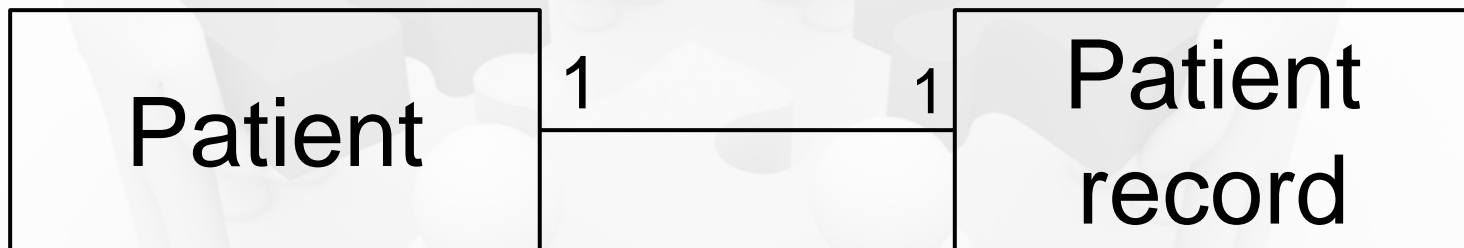
Асоциации

Например – начин за представяне на атрибутите на класовете



UML диаграмите в примери

Асоциация

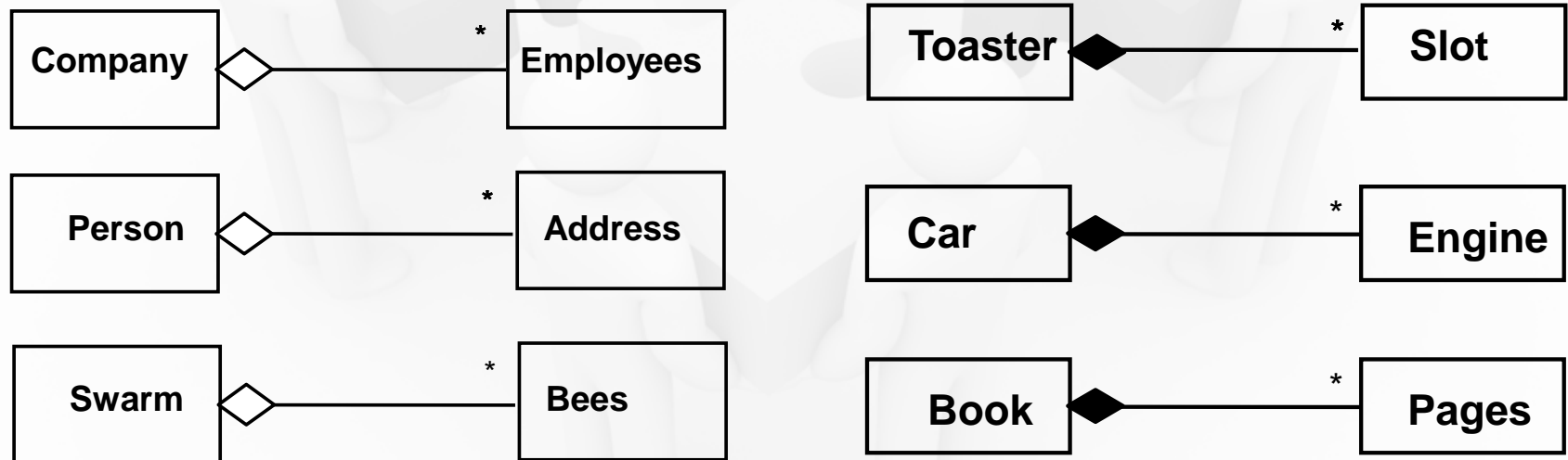


Други видове асоциации

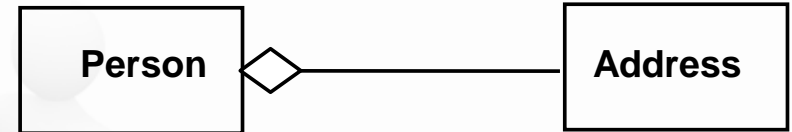
- Агрегация (*Aggregation*)
 - Асоциация, която представя взаимоотношение между „цялото“ и съставлящите го части от типа – (*has-a*) т.е. отделните съставлящи обекти на даден обект може да го съставят но може да съществуват и отделно от него, както може и да са части на други обекти
- Композиция (*Composition*)
 - По-строго взаимоотношение между цялото и съставлящите го части от типа (*part-of*) , т.е. отделните съставлящи обекти се създават заедно с главния обект и не могат да съществуват без него
- Обобщаване (*Generalization*)
 - Наследяване – този тип асоциация съответства на ключовата дума *extends* в *Java*. Взаимоотношение от типа (*is-a*)

Агрегационни модели (Aggregation models)

Показват по какъв начин класовете са съставени от други класове



Агрегация

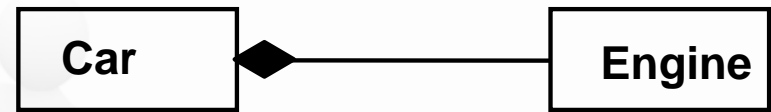


```
public class Address {  
    . . .  
}  
  
public class Person {  
    private Address address;  
    public Person(Address address) {  
        this.address = address;  
    }  
}
```



```
Address address = new Address();  
Person person = new Person(address);
```

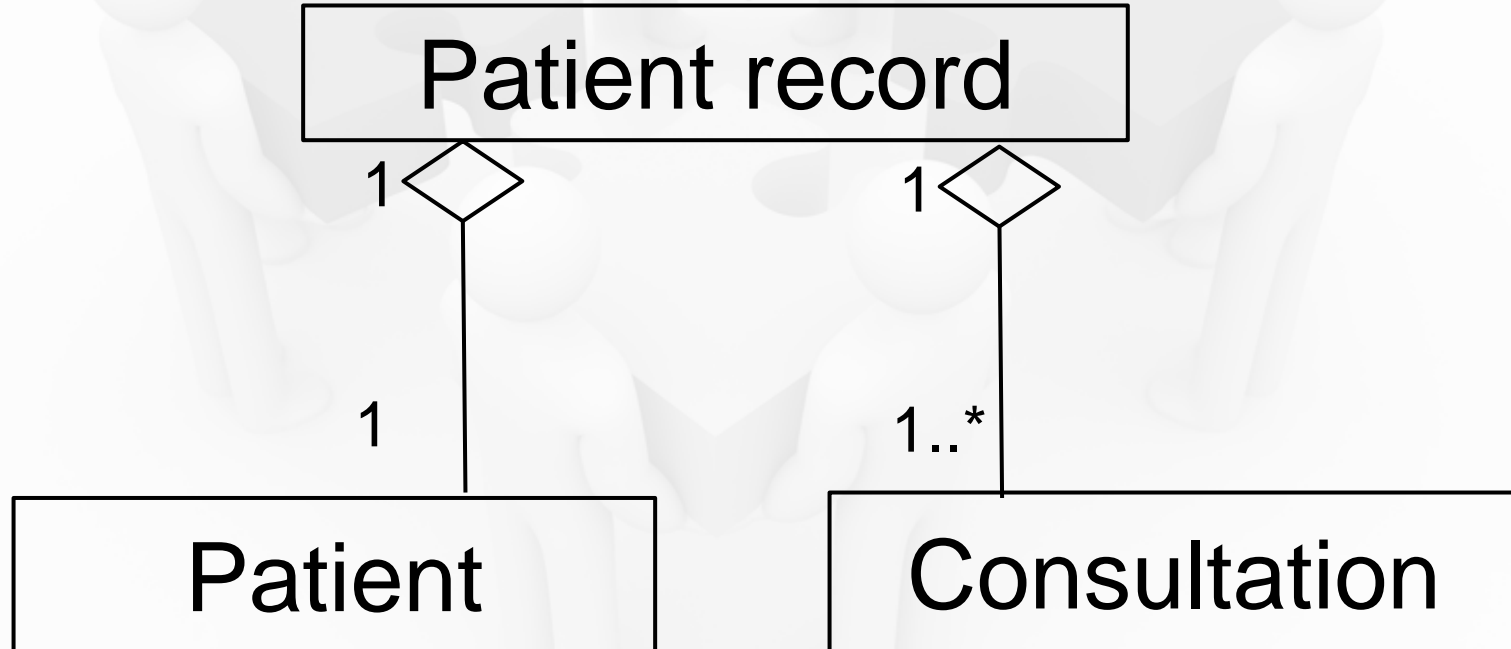
КОМПОЗИЦИЯ



```
public class Engine
{
    . . .
}
public class Car
{
    Engine e = new Engine();
    . . . . .
}
```

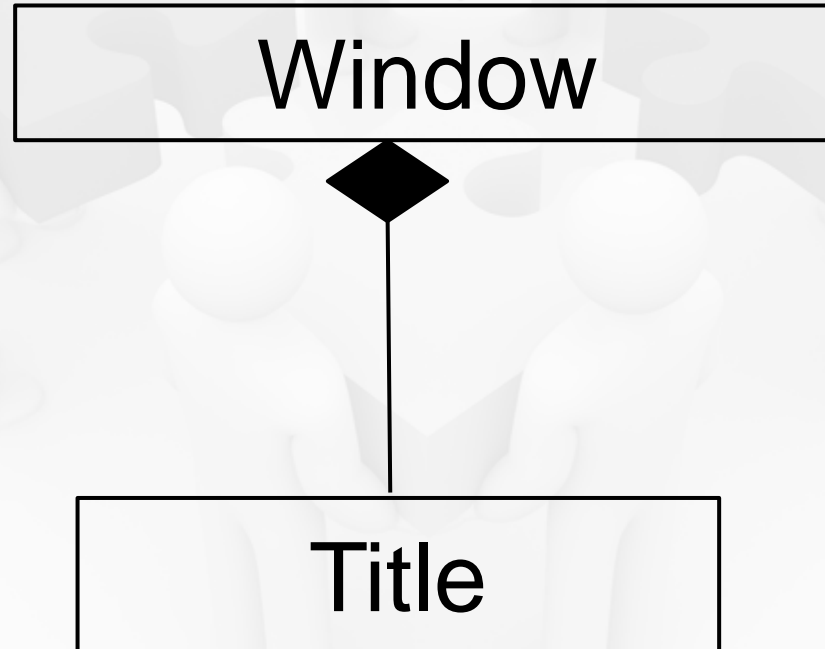
UML диаграмите в примери

Агрегация



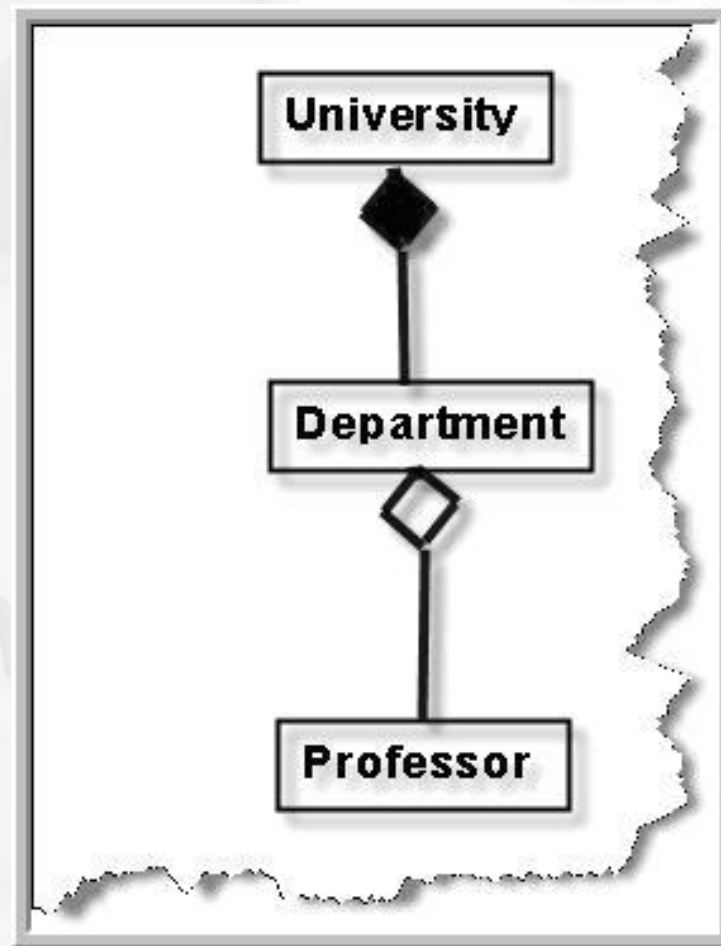
UML диаграмите в примери

Композиция



UML диаграмите в примери

Агрегация и композиция

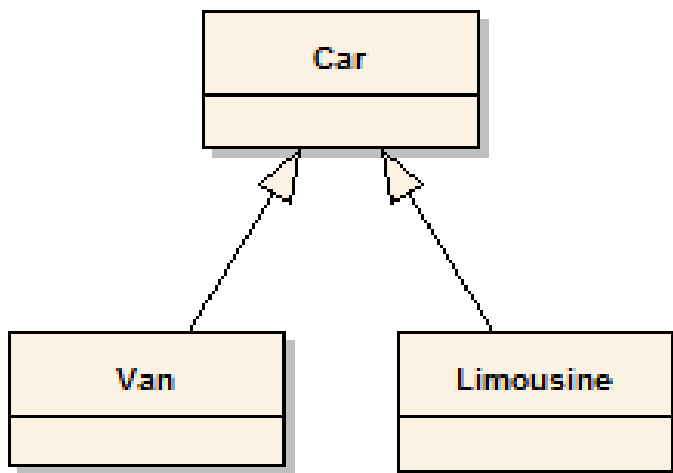


Наследяване

- Обектите са членове на класове, които дефинират свойствата и методите
- Класовете се организират в йерархия, където даден клас (супер-клас), обобщава няколко подкласа
- Подкласовете наследяват свойствата и методите на своя супер-клас, но може да дефинират и свои собствени
- Ежедневен и интуитивен начин за управление на сложността

UML диаграмите в примери

Наследяване



```
class Car {
  ...
}

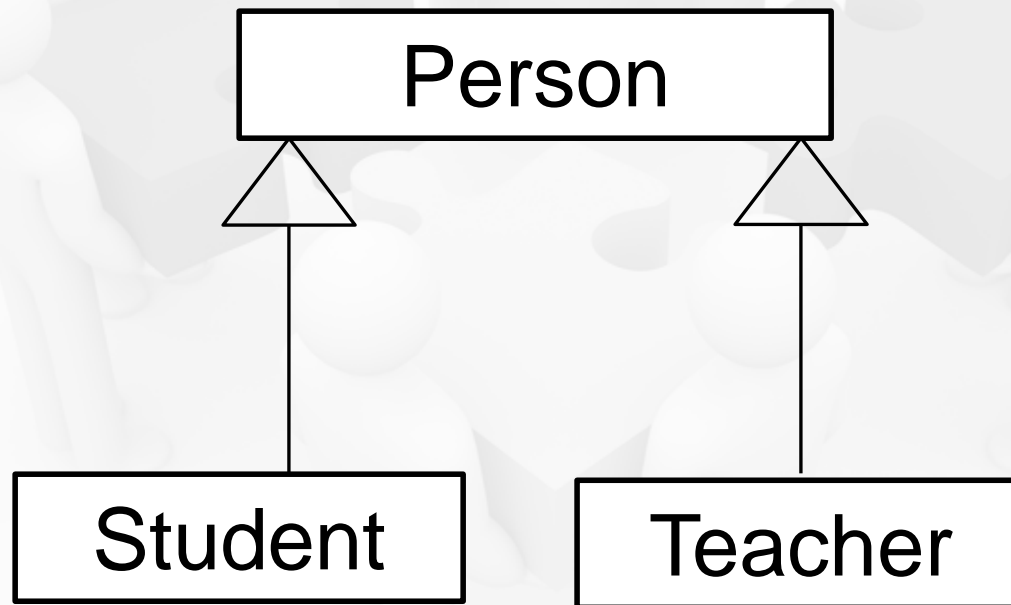
class Limousine extends Car {
  ...
}

class Van extends Car {
  ...
}
```


- Как ще изглежда клас-диаграмата на система за управление на ресурсите в университет
 - Записване/отписване на студенти
 - Проверка за заети зали/запазване на зала
 - Записване на курсове
 - Оптимизация на седмичната програма за всеки студент
 - И т.н.

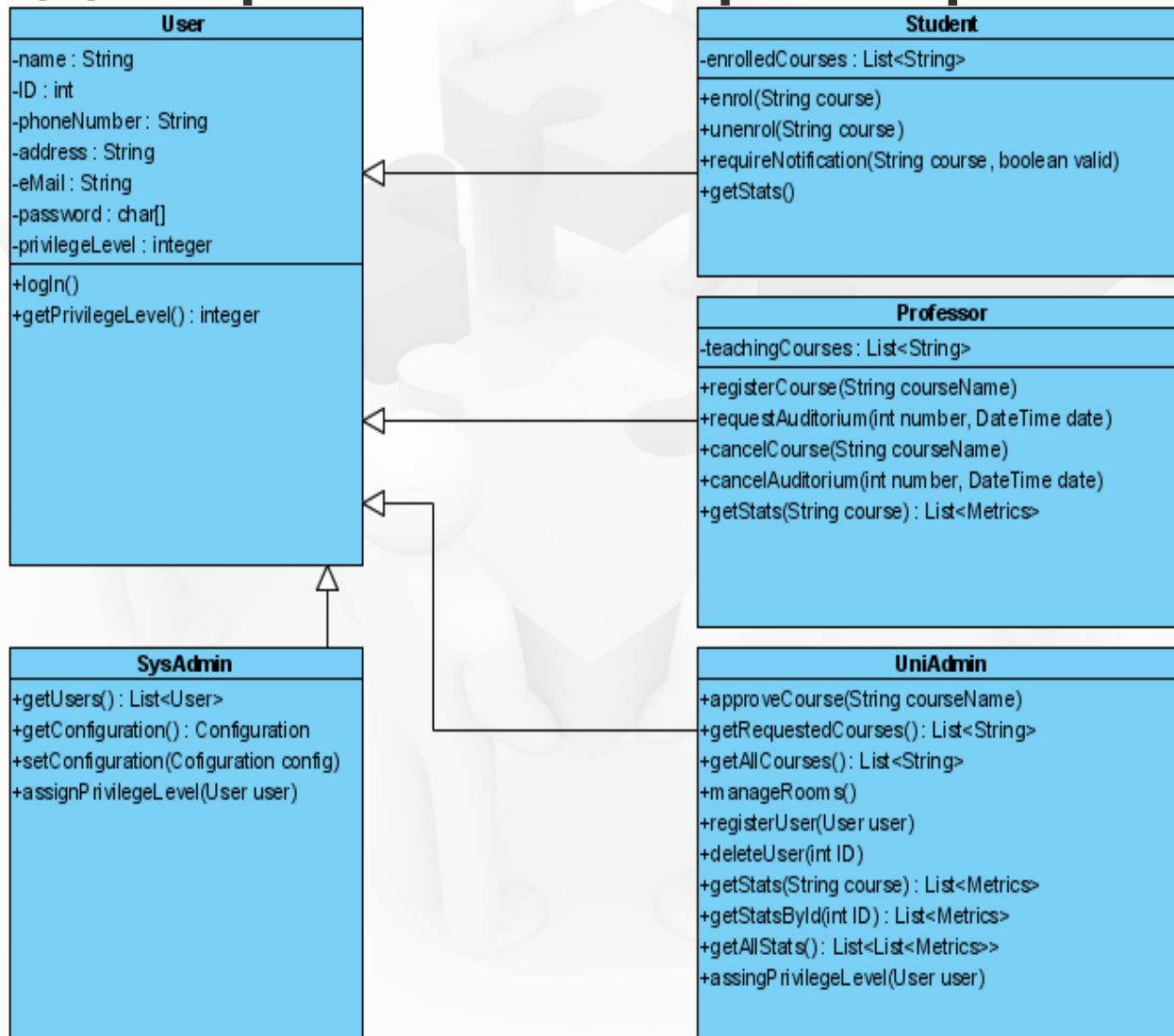
UML диаграмите в примери

Наследяване



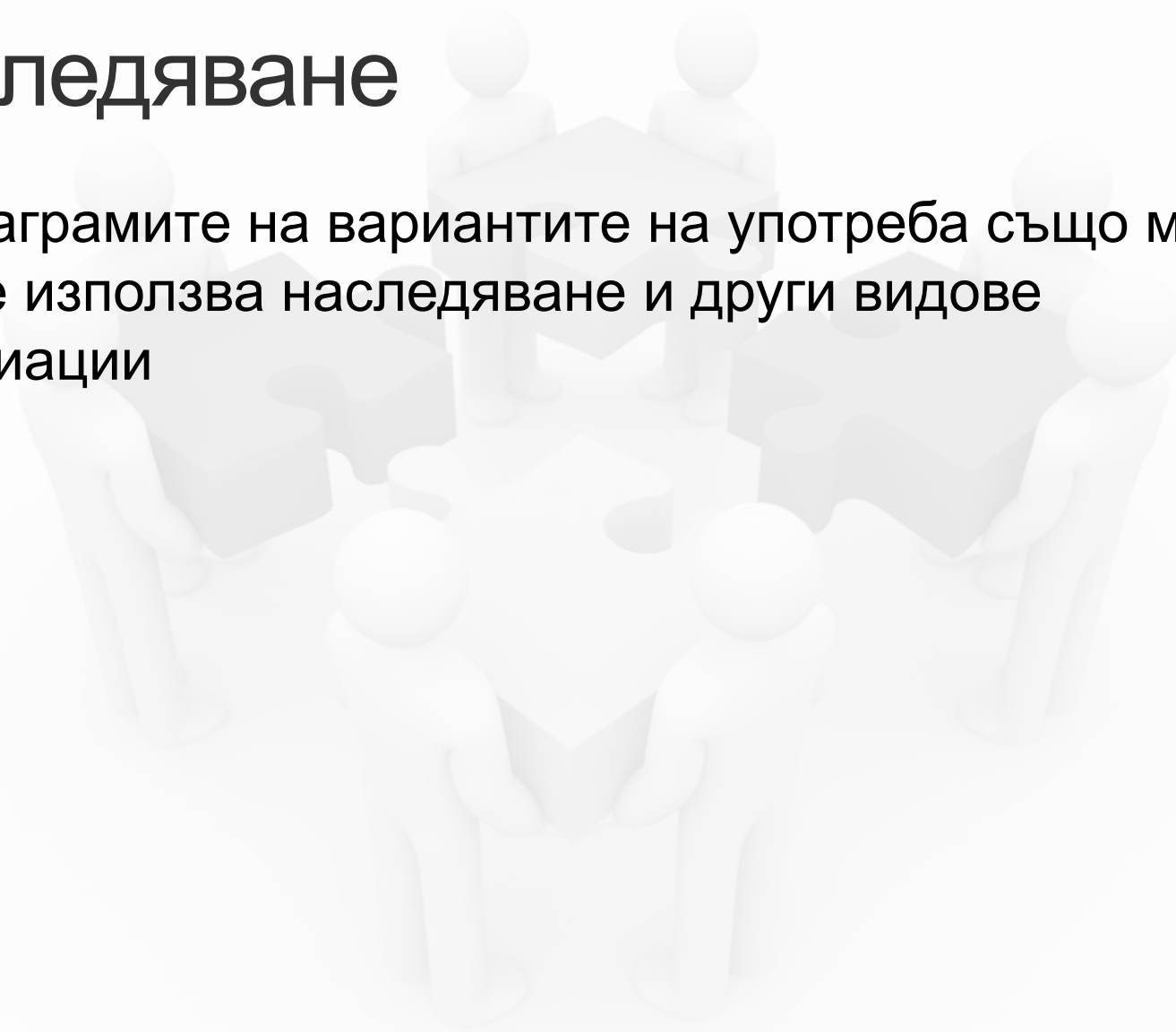
UML диаграмите в примери

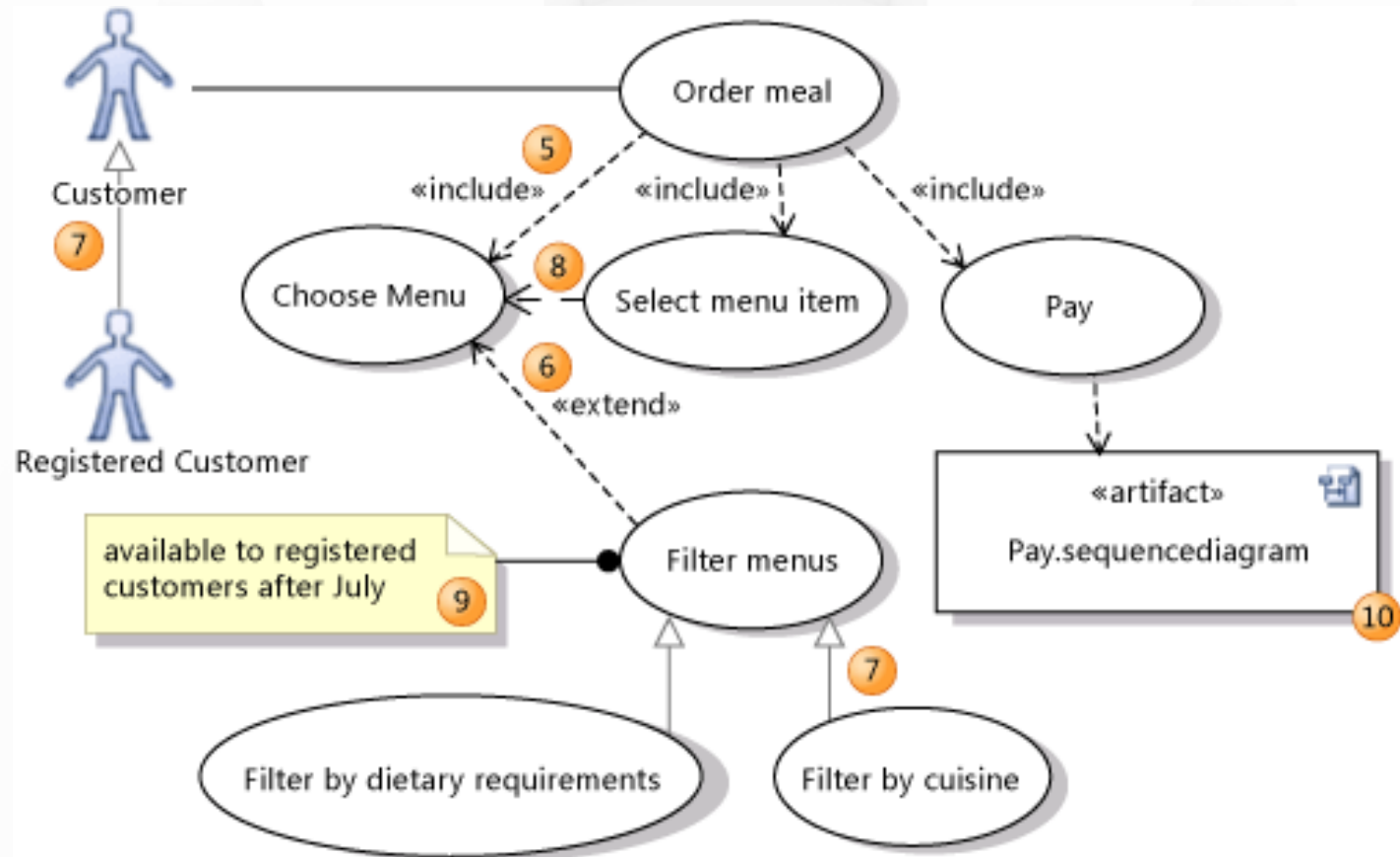
Наследяване



Наследяване

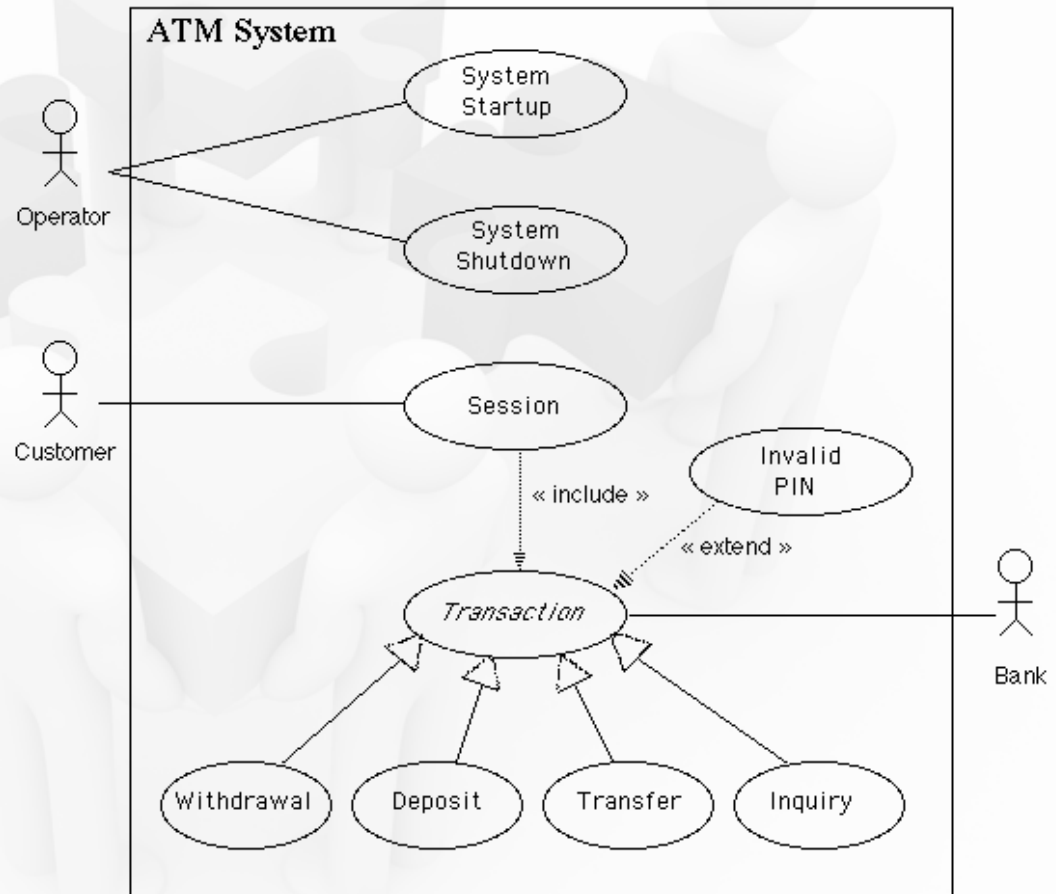
- В диаграмите на вариантите на употреба също може да се използва наследяване и други видове асоциации



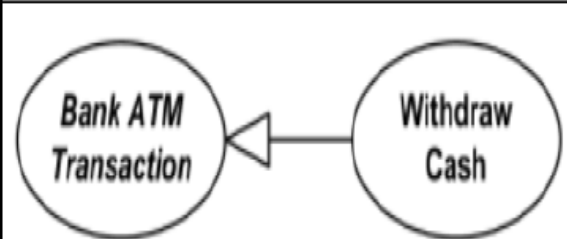
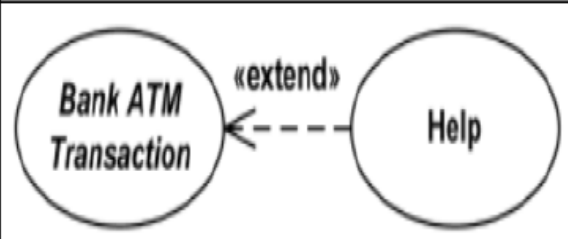
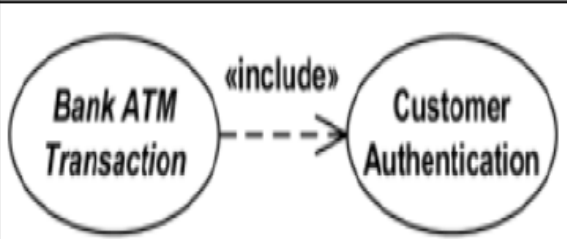


UML диаграмите в примери

Асоциации в Use case диаграмите



Разлики между включване, разширение и генерализация на случаи на употреба

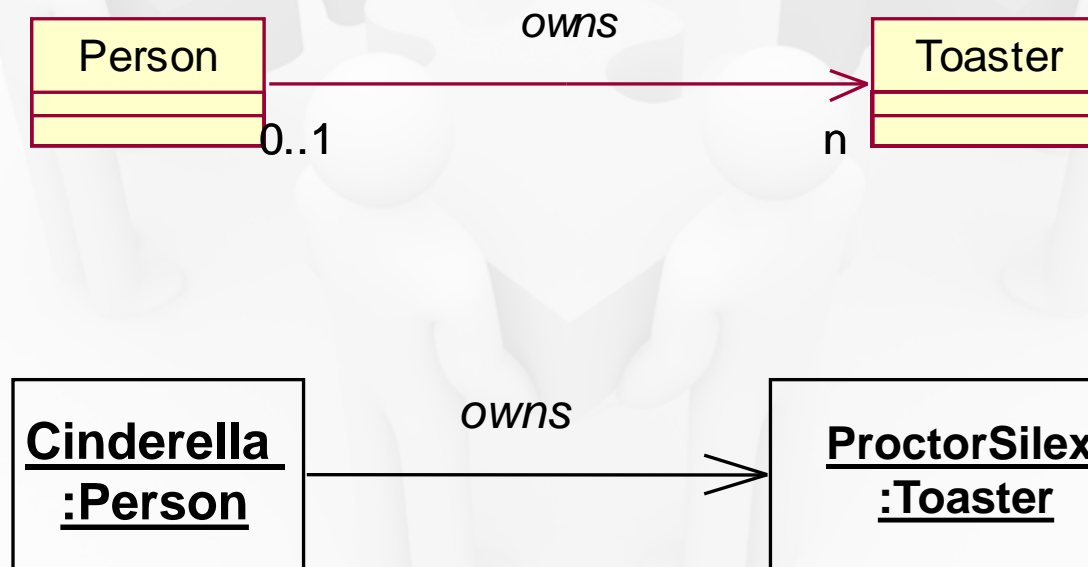
Generalization	Extend	Include
		
Base use case could be abstract use case (incomplete) or concrete (complete).	Base use case is complete (concrete) by itself, defined independently.	Base use case is incomplete (abstract use case).
Specialized use case is required, not optional, if base use case is abstract.	Extending use case is optional, supplementary.	Included use case required, not optional.
No explicit location to use specialization.	Has at least one explicit extension location.	No explicit inclusion location but is included at some location.
No explicit condition to use specialization.	Could have optional extension condition.	No explicit inclusion condition.

Допълнителни елементи на клас-диаграмите

- Стереотип (Stereotypes)
 - Механизъм за разширение на UML
 - Разширява набора от елементи в моделите
 - Показват се като анотации с текст, заграден от двойни ъглови скоби (<<>>)
- Обектни диаграми (Object Diagrams)
 - Показват конкретни екземпляри на класовете
 - Наричат се още диаграми на екземплярите (Instance Diagram)
 - Имената на обектите са от вида: *instance name* : *class name*.
- Представяне на интерфейси и абстрактни класове

UML диаграмите в примери

Обектни диаграми

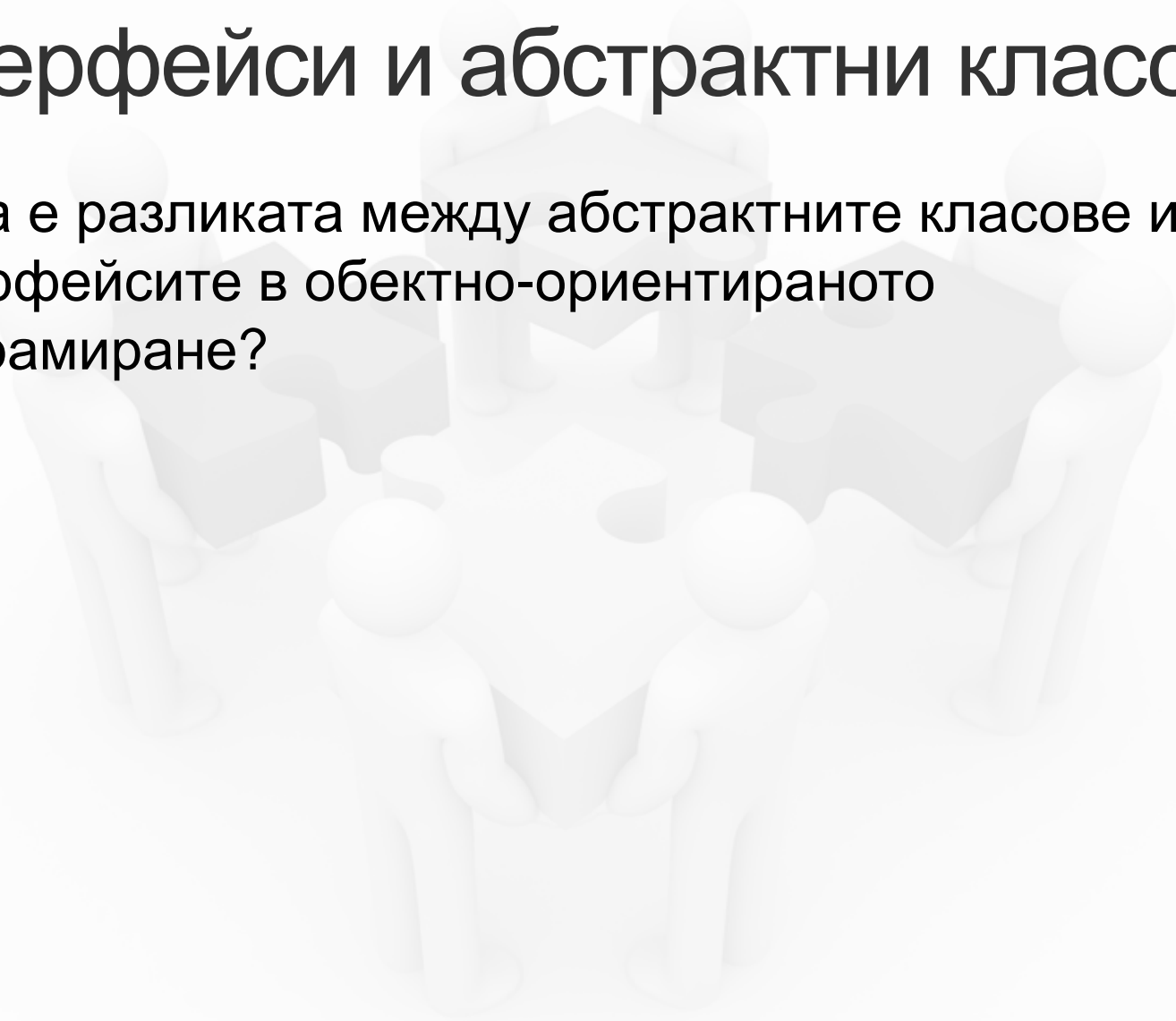


Interfaces and Abstract Classes

- UML Interface
 - A class that has only public operations with no method bodies or attributes.
- Representation
 - Italicize the name of interfaces or abstract classes
 - Use the {abstract} constraint for abstract classes (optional) and the <<interface>> stereotype for interfaces
- Realization
 - Relationship between an implementation class and an interface
 - Drawn with a dashed line and a white triangular arrow touching the interface

Интерфейси и абстрактни класове

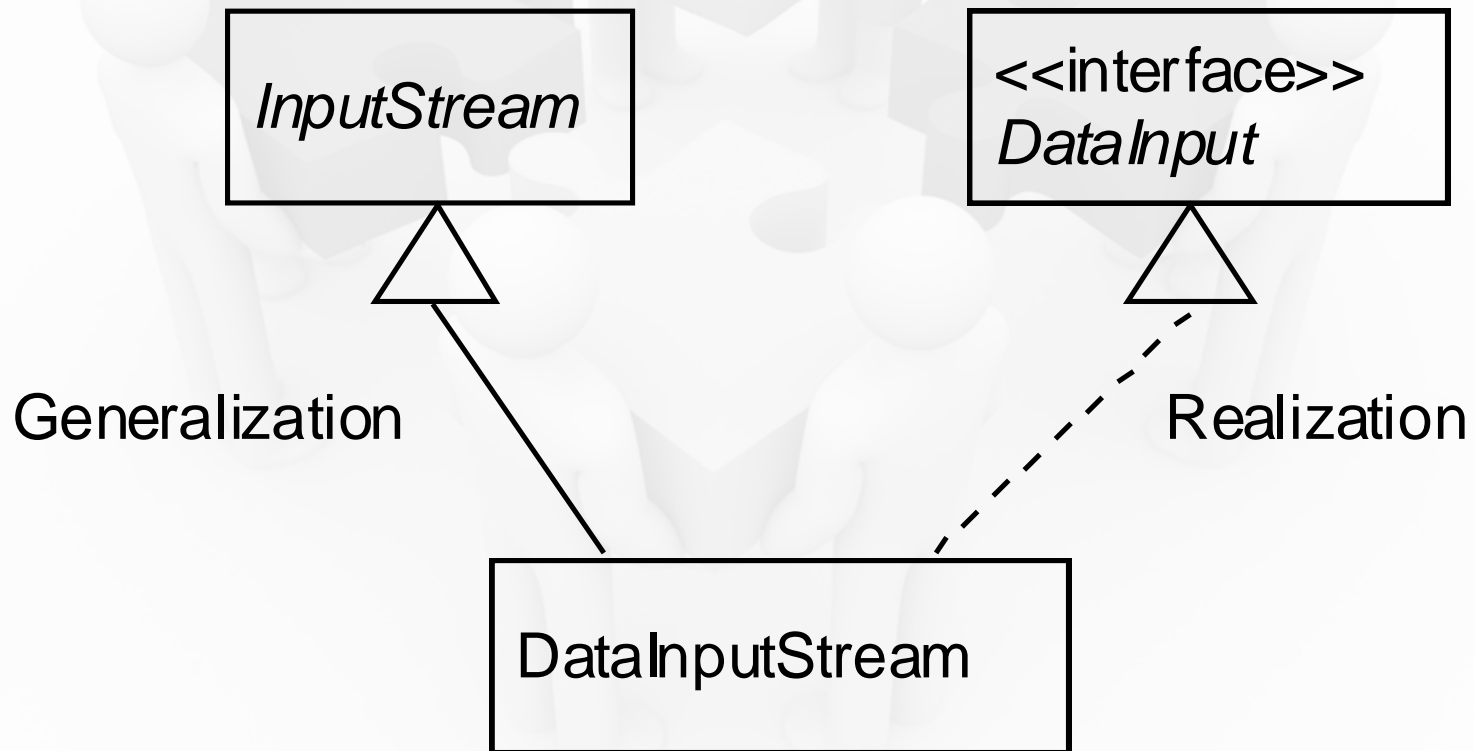
- Каква е разликата между абстрактните класове и интерфейсите в обектно-ориентираното програмиране?



UML диаграмите в примери

Наследяване на клас

Реализация на интерфейс



Клас диаграми - обобщение

this class is associated with



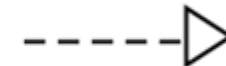
this class

this class inherits from



this class

this class is a realization of



this interface

these classes compose without belonging to



this class

these classes compose and are contained by



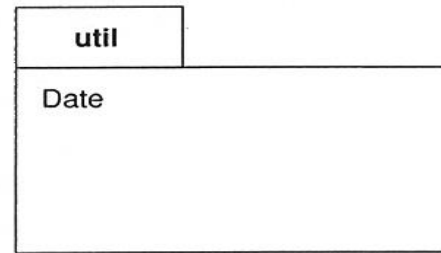
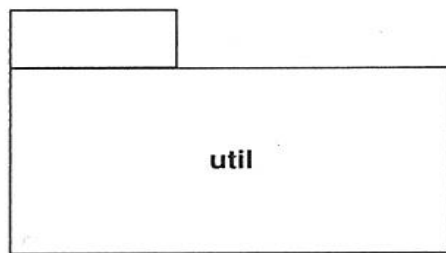
this class

Пакетни диаграми (Package Diagrams)

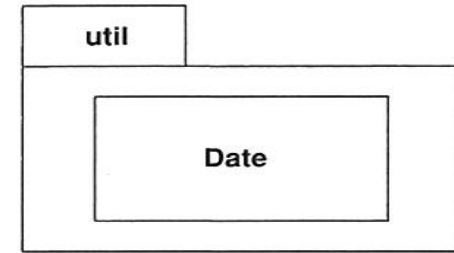
- Пакетите служат за групиране на подобни елементи
 - Най-често срещани са в клас-диаграмите, но може да се използват и с други UML диаграми
 - Използват се за създаване на йерархия и повишаване на нивото на абстракция
- Съответстват на пакетите в *Java* и пространствата от имена (namespace) в C++

UML диаграмите в примери

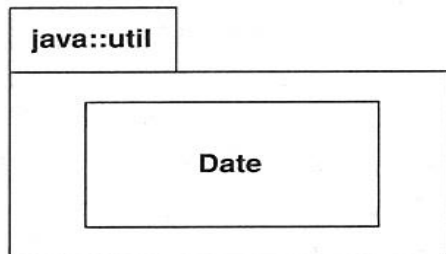
Пакетни диаграми



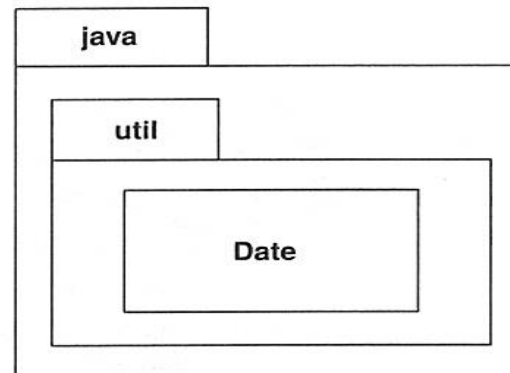
Contents listed in box



Contents diagrammed in box



Fully qualified package name



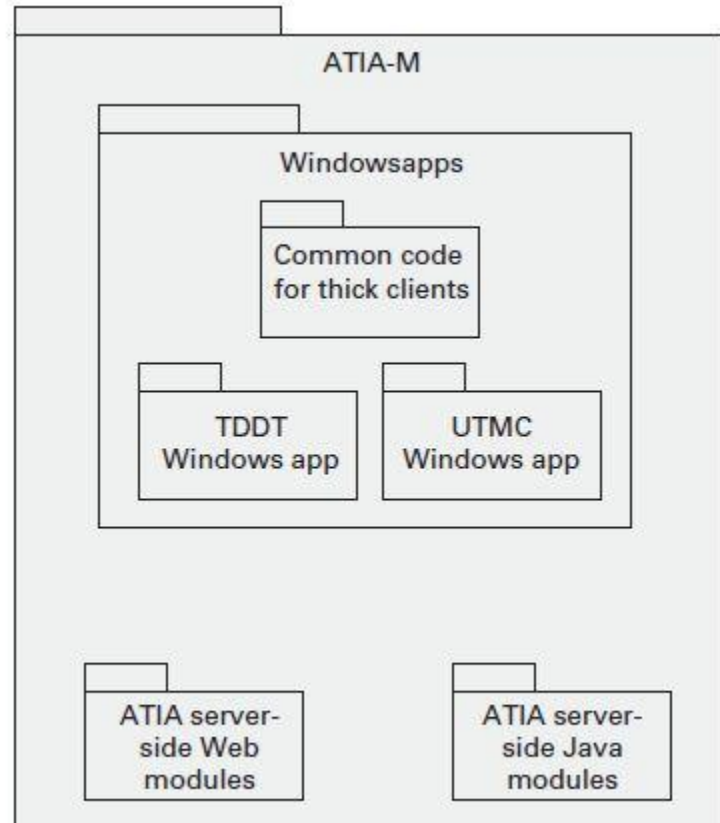
Nested packages



Fully qualified class name

UML диаграмите в примери

Пакетни диаграми



Paul Clements, et al., Documenting
Software Architectures: Views and Beyond,
Addison Wesley 2010

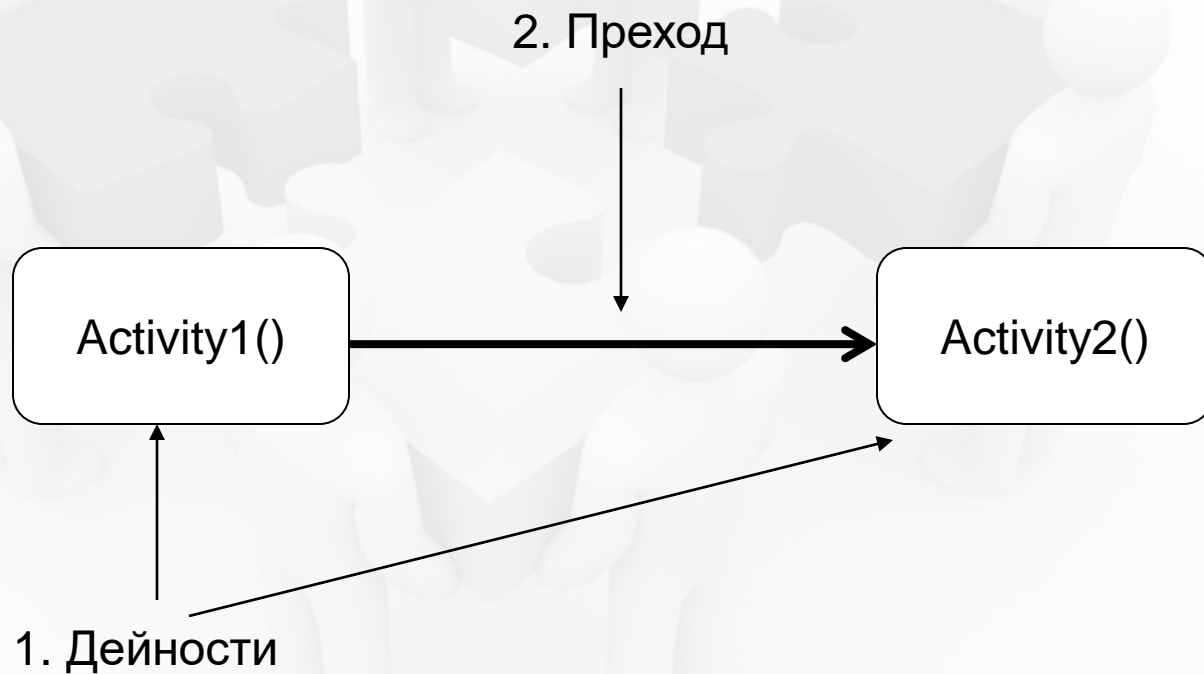
Модели на процесите

- С клас диаграмите се моделира статичната страна на системите
- За динамиката на системата и нейното поведение се използват други диаграми
- Диаграмите на активностите дават възможност да се моделира как системата се използва в рамките на по-обща бизнес процеси

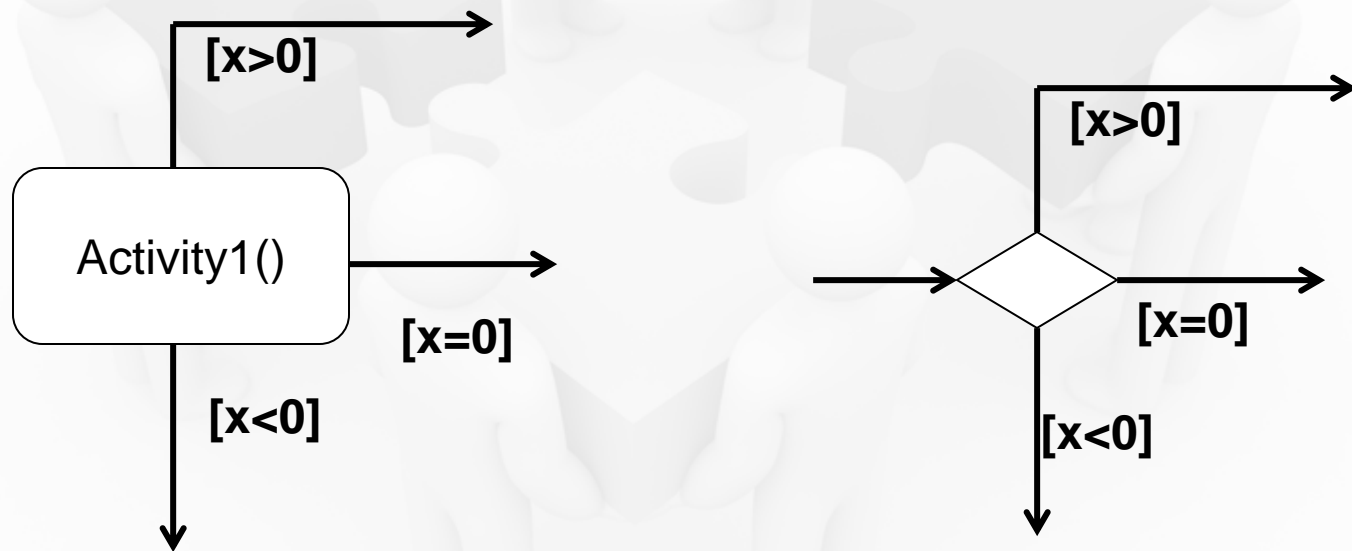
Activity Diagram

- Describes how activities are coordinated.
- Is particularly useful when you know that an operation has to achieve a number of different things, and you want to model what the essential dependencies between them are, before you decide in what order to do them.
- Records the dependencies between activities, such as which things can happen in parallel and what must be finished before something else can start.
- Represents the workflow of the process.

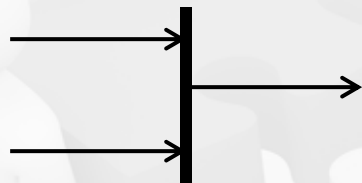
Нотация



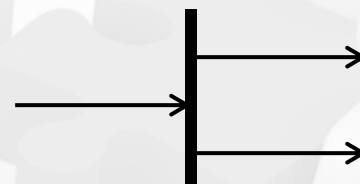
Нотация



Нотация



Обединение на процеси (Join)



Паралелни процеси (Fork)

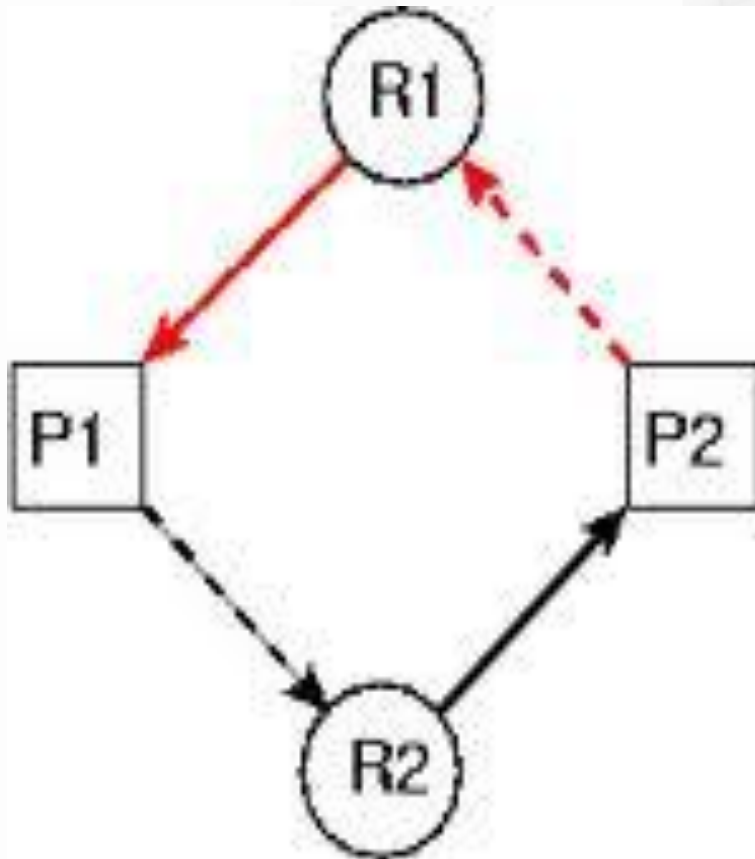
Нотация



UML диаграмите в примери



Мъртва хватка (deadlock)



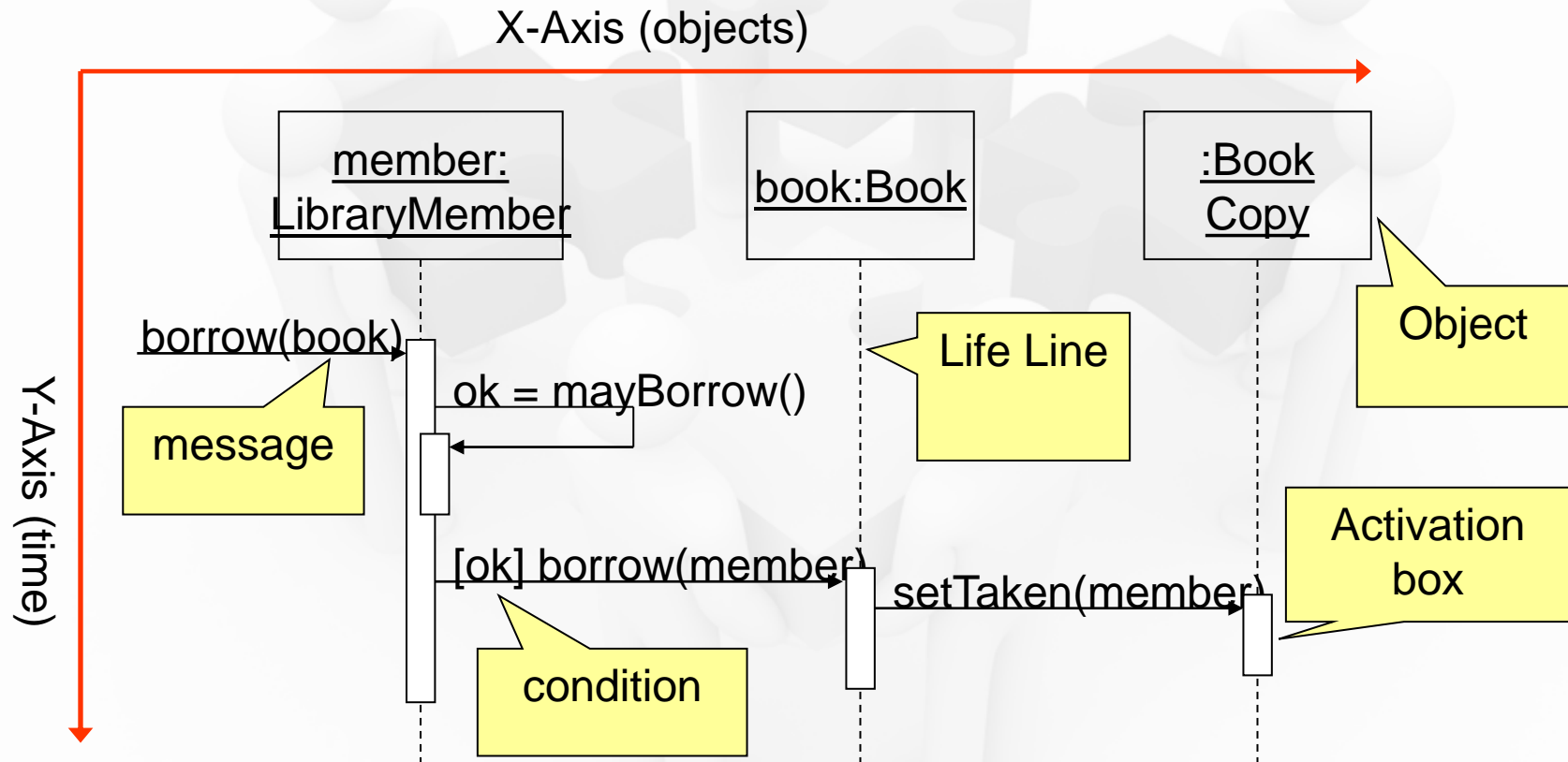
- R1 is held by
- - → is waiting for R1
- R2 is held by
- - → is waiting for R2

Последователностни модели

- Sequence diagrams
- Показват последователността от взаимодействия между обектите
 - Обектите се подреждат хоризонтално
 - Времето се показва по вертикалата с времеви линии за всеки обект
 - Взаимодействията между обектите се представят чрез стрелки
 - Когато даден обект е активен, това се означава с правоъгълник във времевата линия на обекта

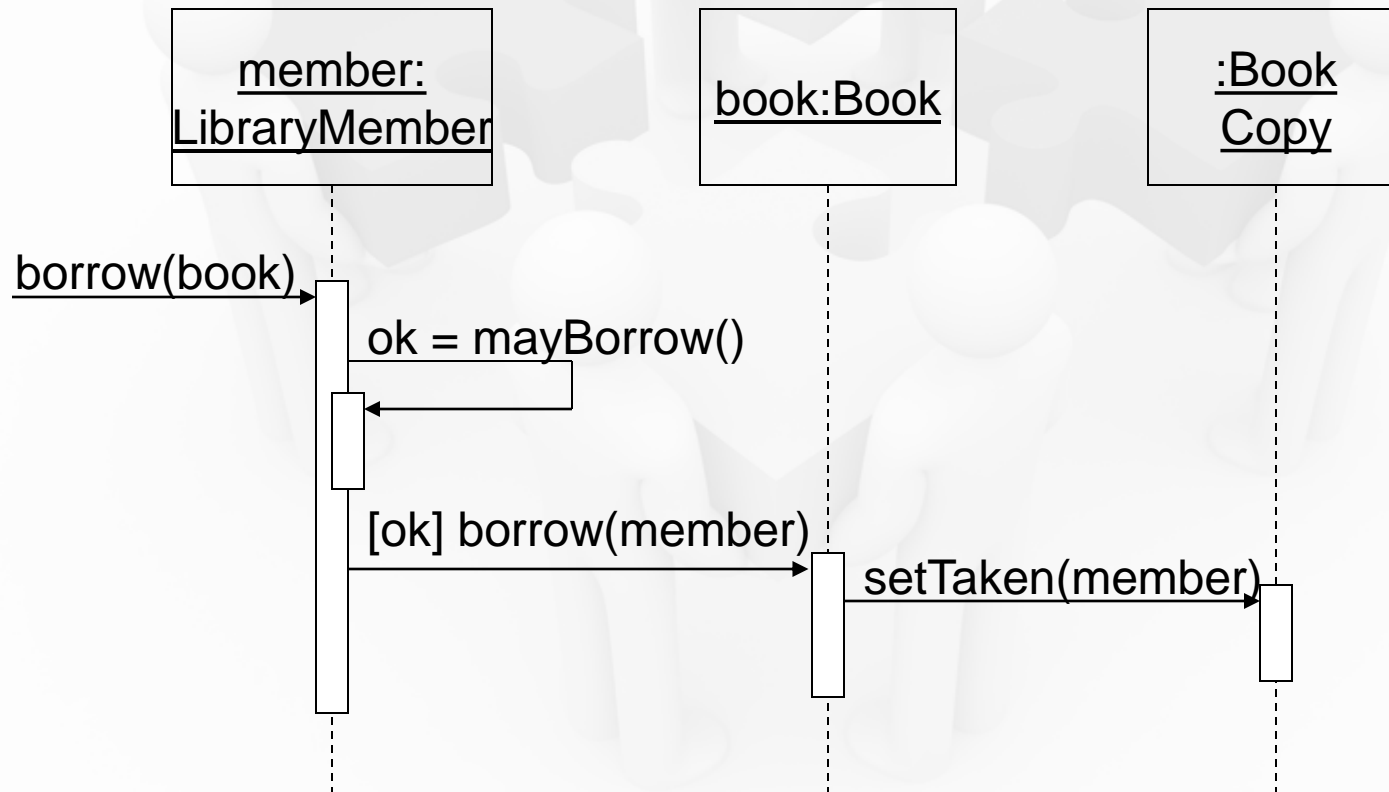
UML диаграмите в примери

Последователностни диаграми



UML диаграмите в примери

Последователностни диаграми



Съобщения (Messages)

- Взаимодействието между два обекта се показва като „съобщение“, изпратено от единия обект към другия (извикване на метод, известяване за събитие и т.н.)
- Ако обектът *obj1* изпрати съобщение към обекта *obj2*, в последователностната диаграма, това се показва с насочена стрелка между тях



- Възможни са примки (съобщения към „себе си“)
- Като минимум стрелката се аотира с името на съобщението. Може да се добавят допълнително параметри, логически условия, цикли и др.

Връщани стойности

- Показват се на диаграмата само ако е необходимо
 - Не се препоръчва да се показват, ако върнатия резултат е очевиден и е нужно да се покаже на друго място (например като параметър на следващо съобщение)
- За да се подобри четивността се използват прекъснати линии

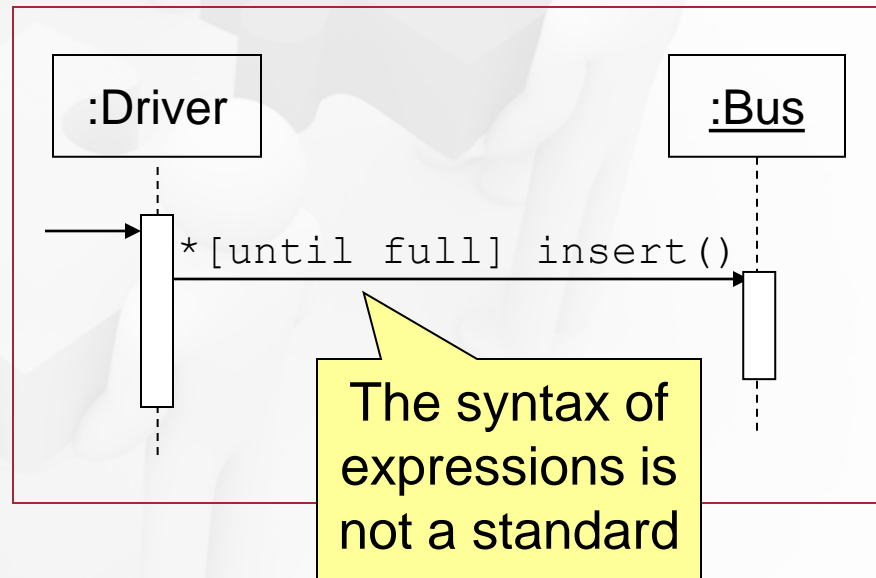
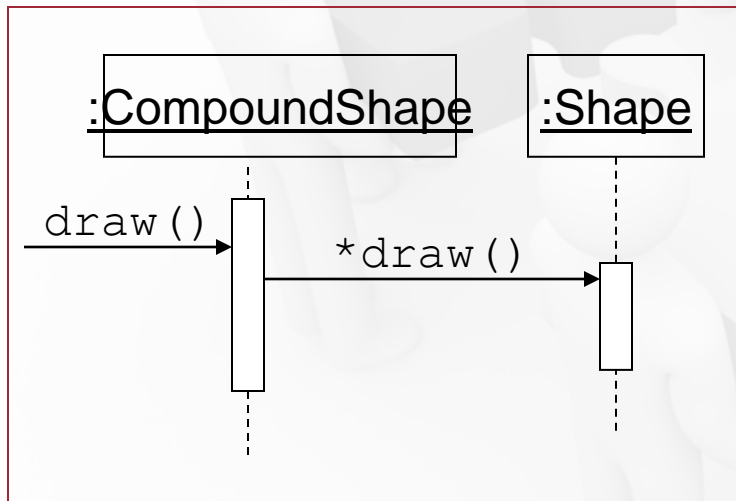


Допълнителна информация

- Условие
 - Синтаксис: '[' expression ']' message-label
 - Съобщението се изпраща, само ако условието има стойност логическа истина true
 - Пример: [ok] borrow(member)
→
- Цикъл
 - Синтаксис: * ['[' expression ']'] message-label
 - Съобщението се изпраща многократно

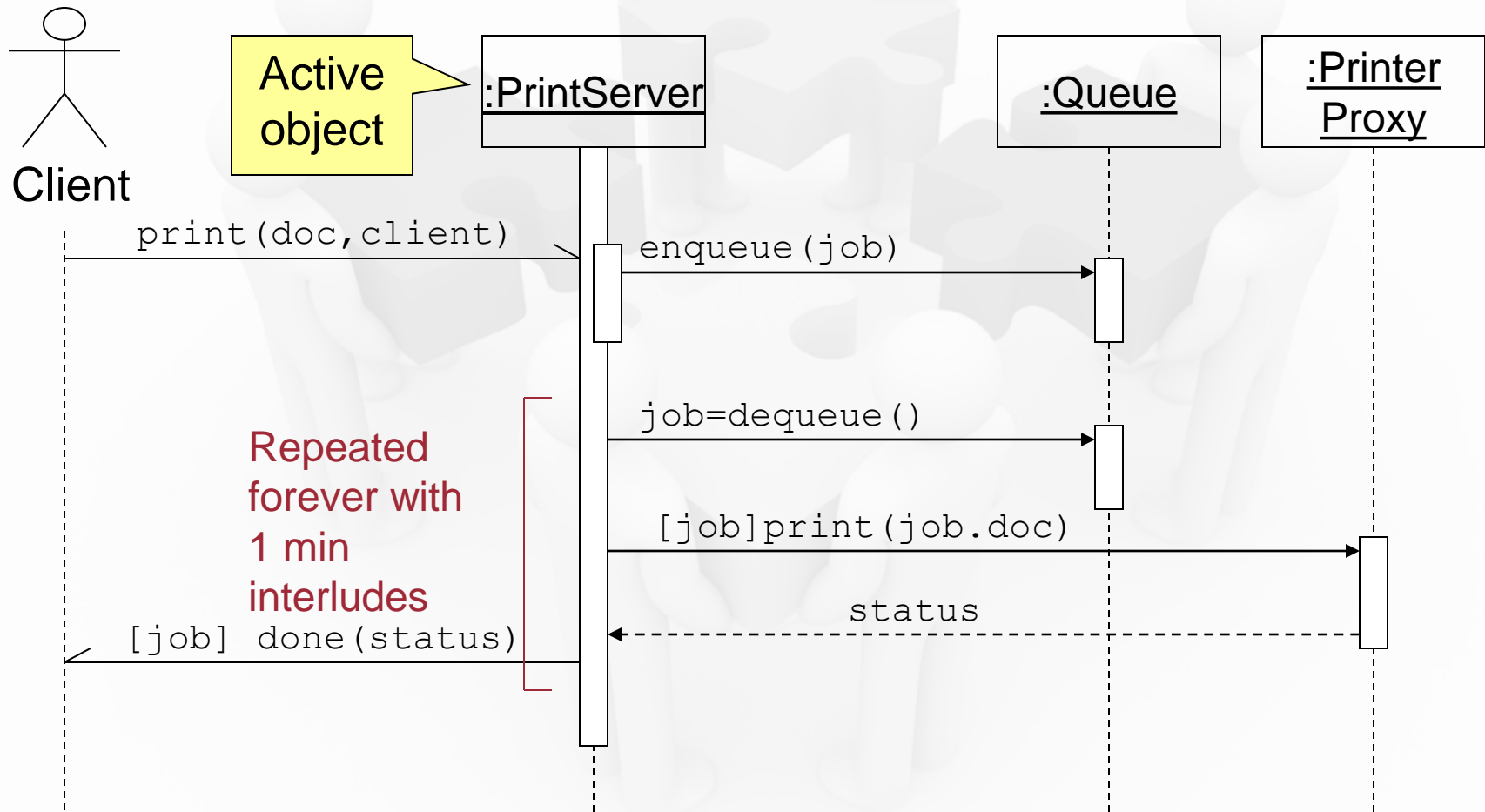
UML диаграмите в примери

Цикъл в последователностните диаграми



UML диаграмите в примери

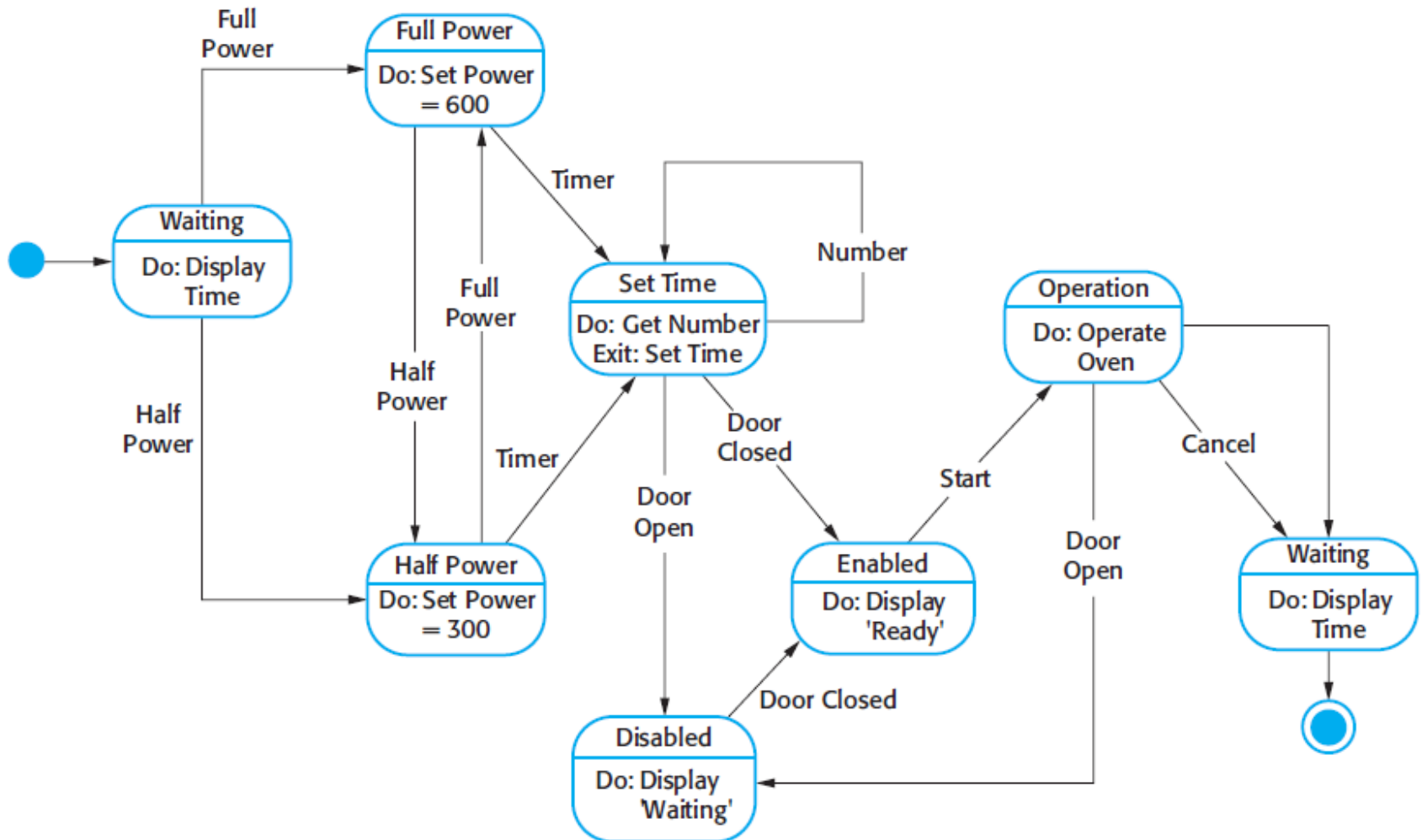
Printing A Document



Диаграми на състоянията

- Statecharts в UML
- Моделират поведението на системата в отговор на различни външни или вътрешни събития
- Често се използват за моделиране на системи за реално време
- Представяват граф, като възлите на графа моделират различни състояния на системата, а дъгите – събитията, които водят до преход от едно състояние в друго
- Показват как обектите отговарят на различни заявки и промените на състоянието в следствие на тези заявки

Диаграма на състоянията на микровълнова печка



Състояния и събития на диаграмата

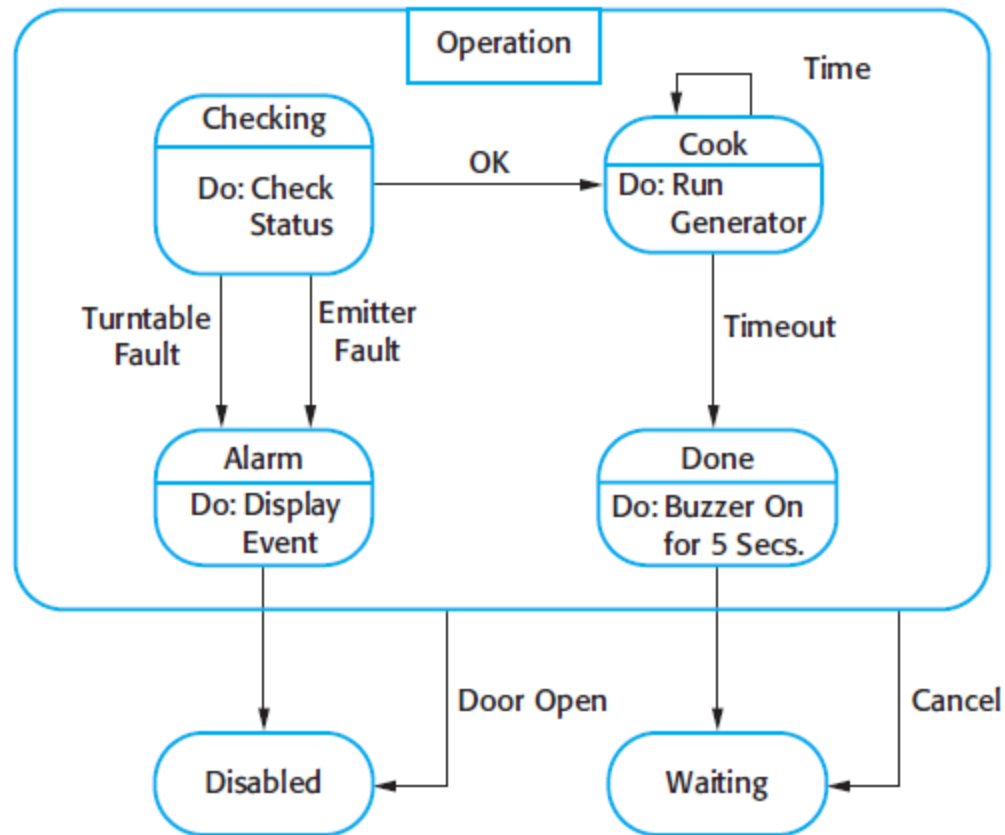
State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

СЪСТОЯНИЯ И СЪБИТИЯ НА ДИАГРАМАТА

Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

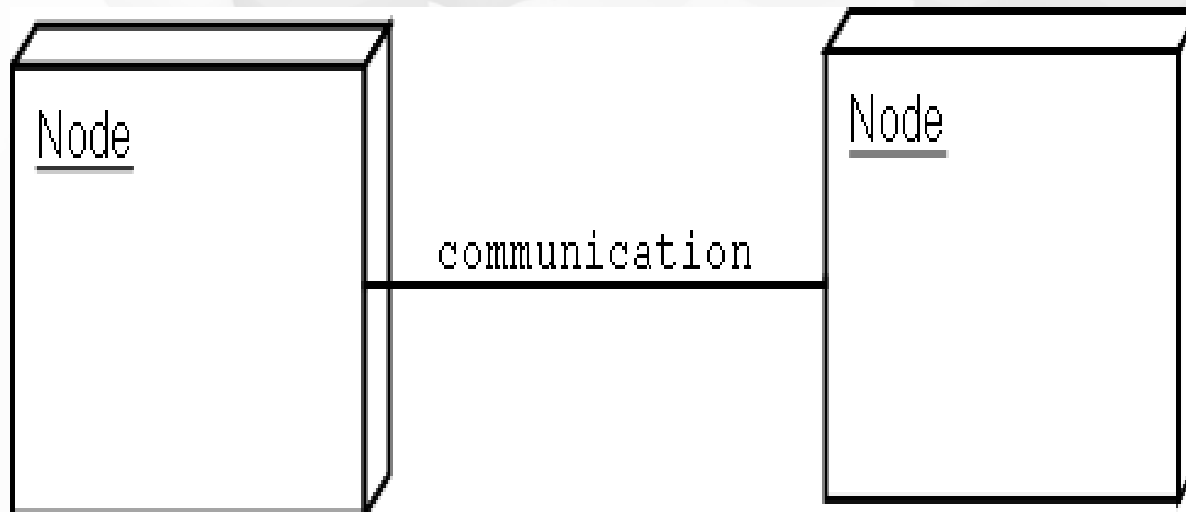
- За да се избегне взрив на състоянията при големи процеси може да се въведе абстракция на някои състояния, като те се покажат разширено на отделна диаграма
- На следващия слайд е показана Statechart която описва допълнително състоянието “*operation*” на микровълновата печка

Microwave oven operation

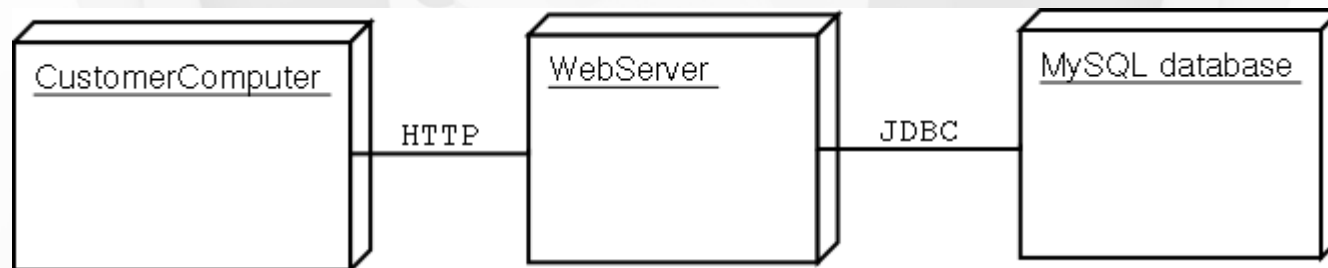


Deployment Diagram

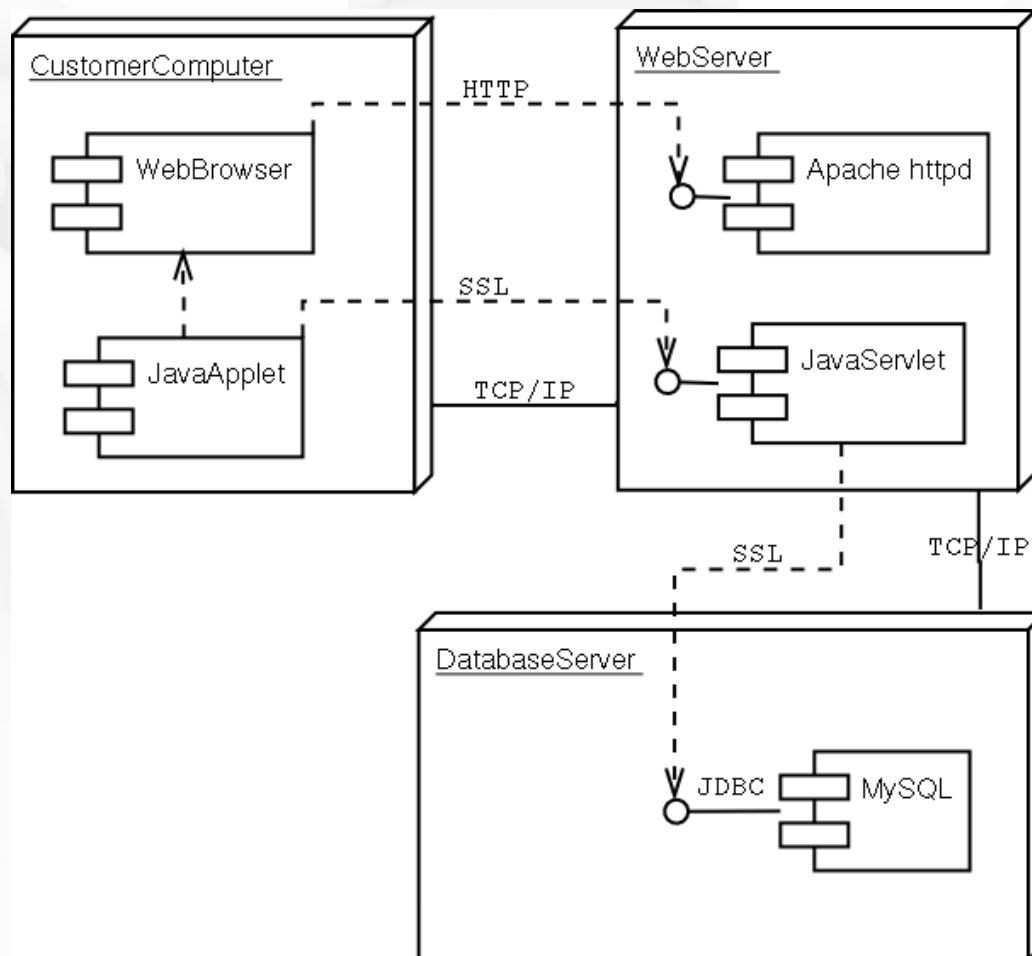
- Представя връзките между хардуерни и софтуерни елементи в готовата система



Deployment Diagram



Пример

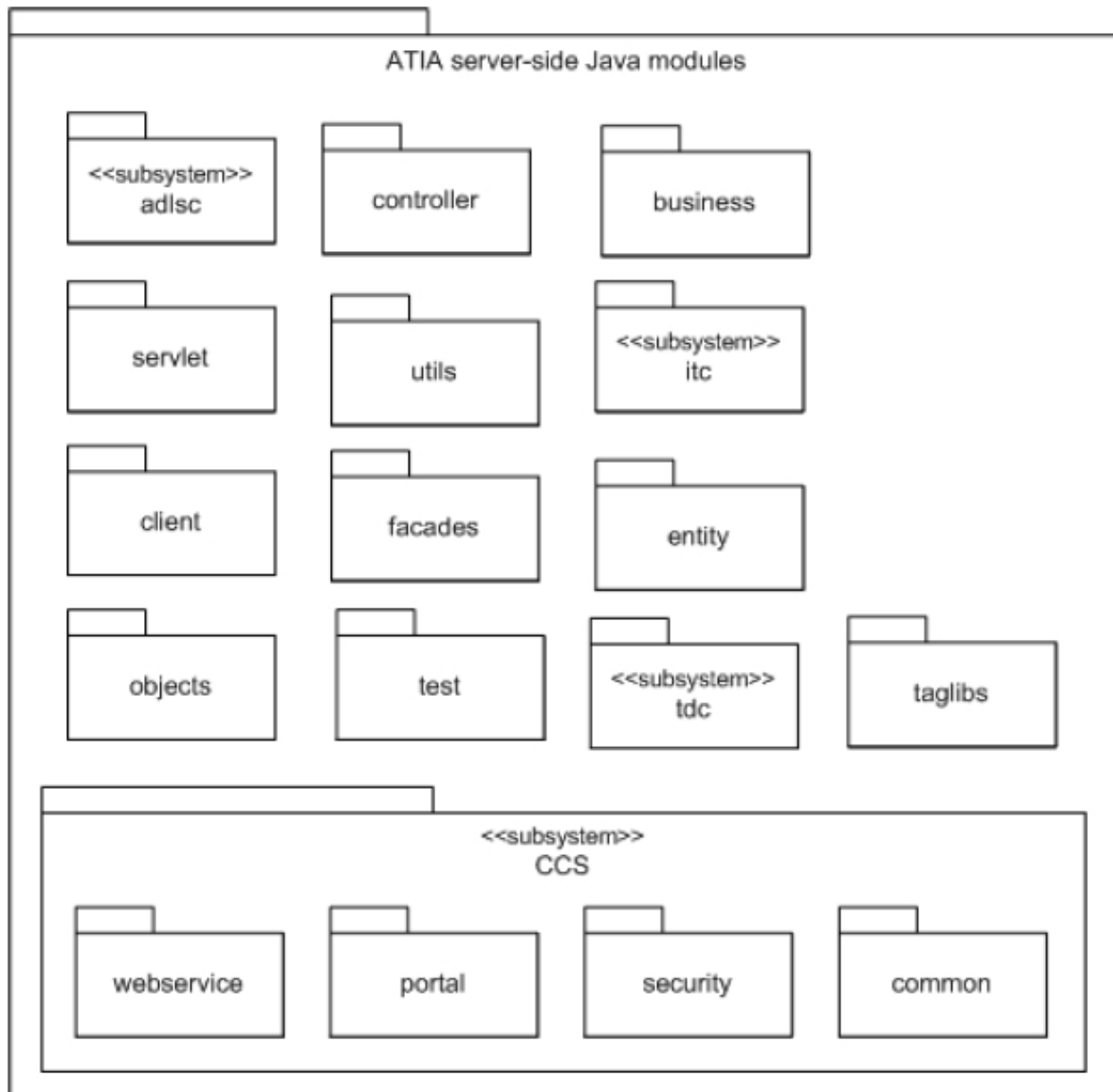


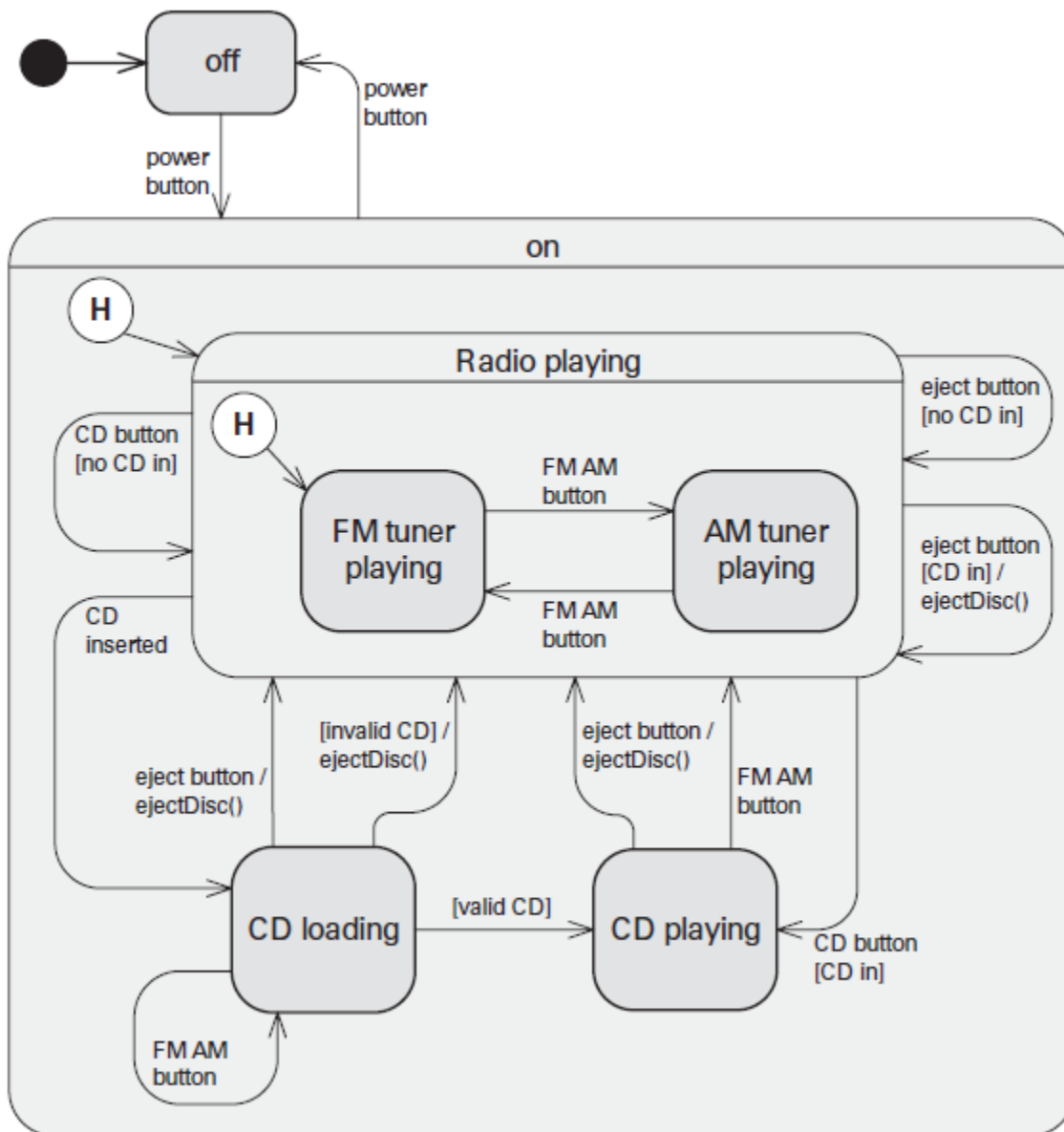
Обобщение

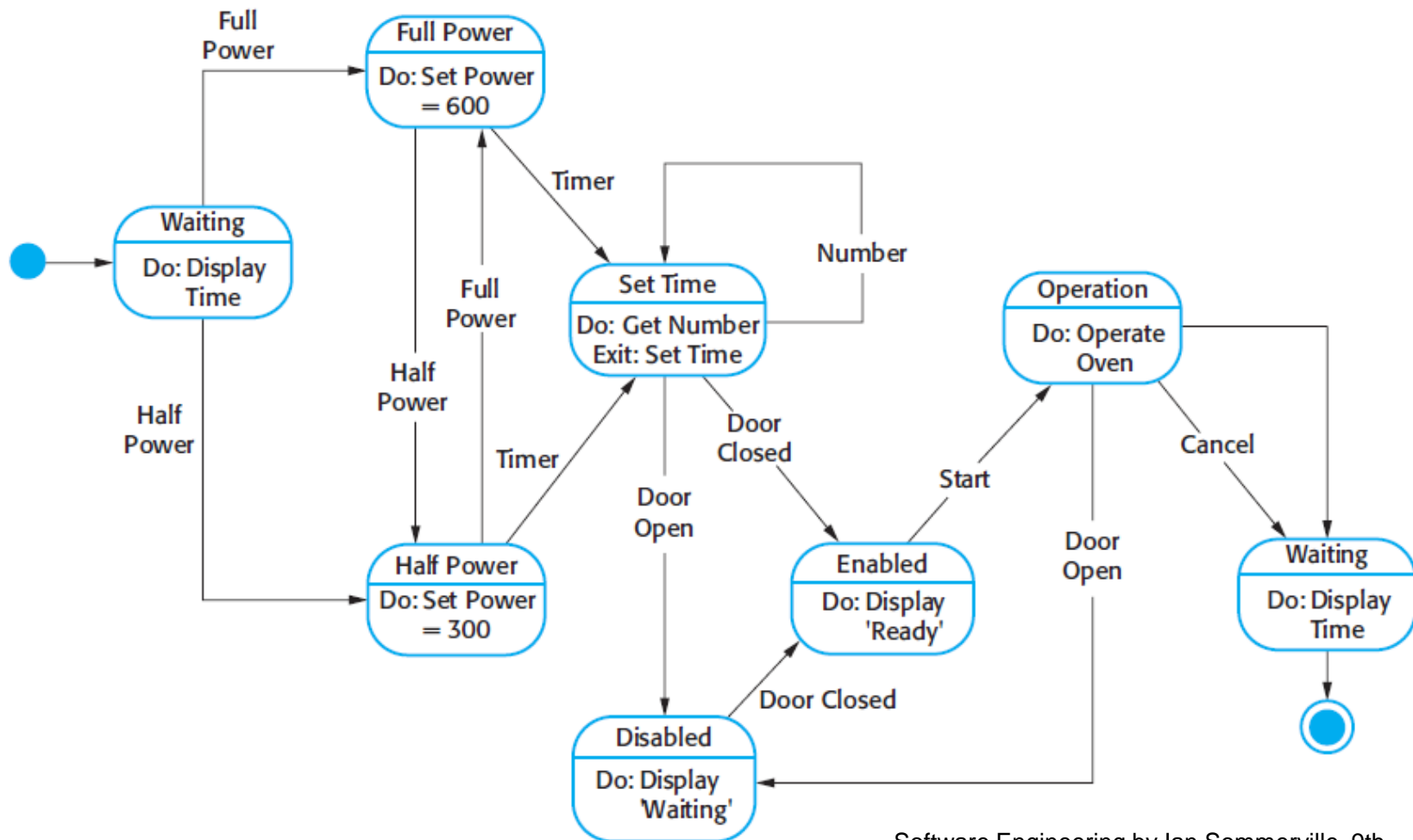
- Статични модели
 - Клас диаграми
 - Пакетни диаграми
 - И др...
- Модели на поведението
 - Диаграми на активностите (activity)
 - Диаграми на състоянията (state)
 - Диаграми на вариантите на употреба (use case)
 - Модели на взаимодействията
 - Диаграми на последователността (Sequence)
 - И др...

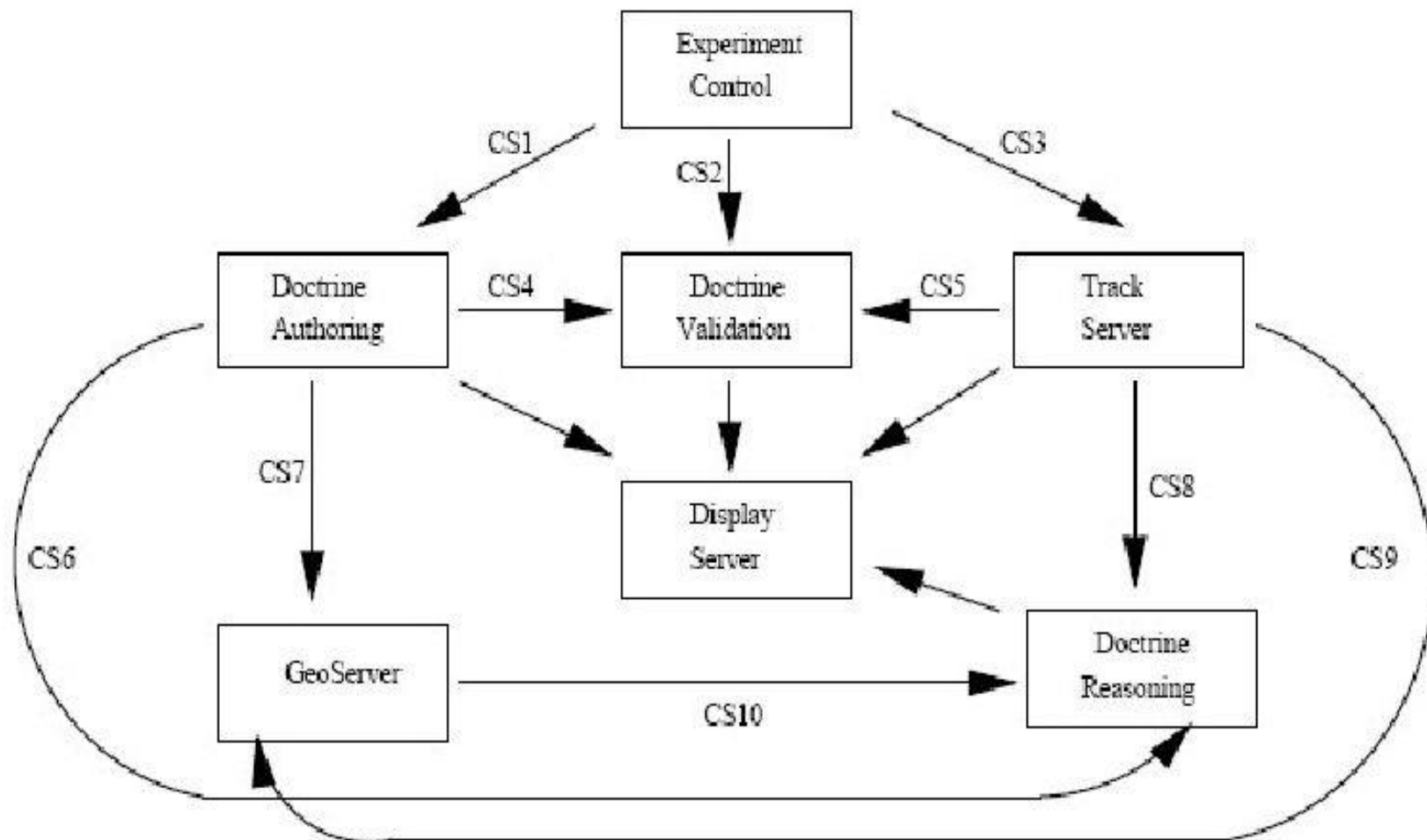
Отново за архитектурата

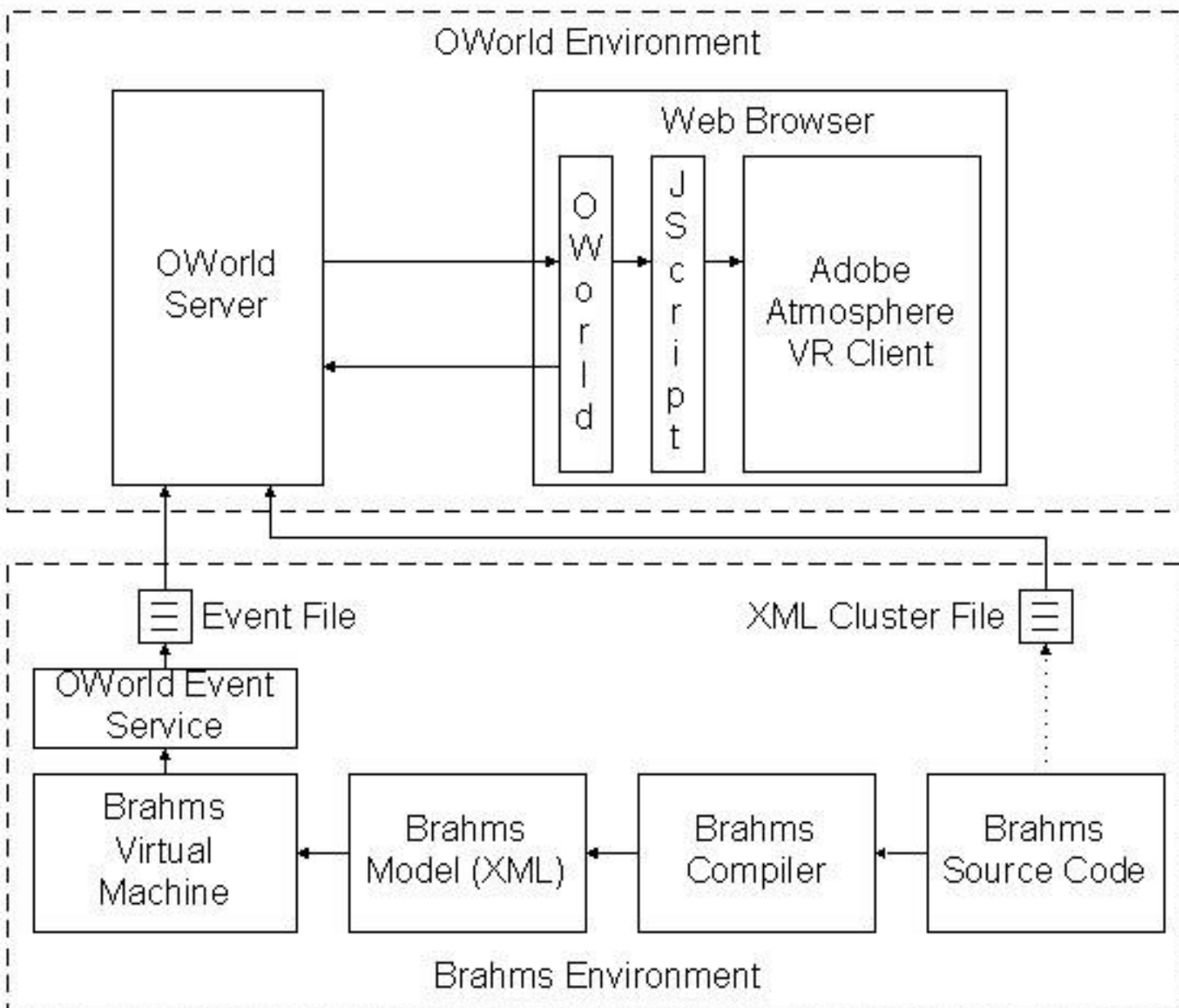
- Съвкупност от различни изгледи на системата, които се състоят от софтуерни елементи, външно видимите характеристики на тези елементи, и връзките, които съществуват между тях
- А какво означава изглед?

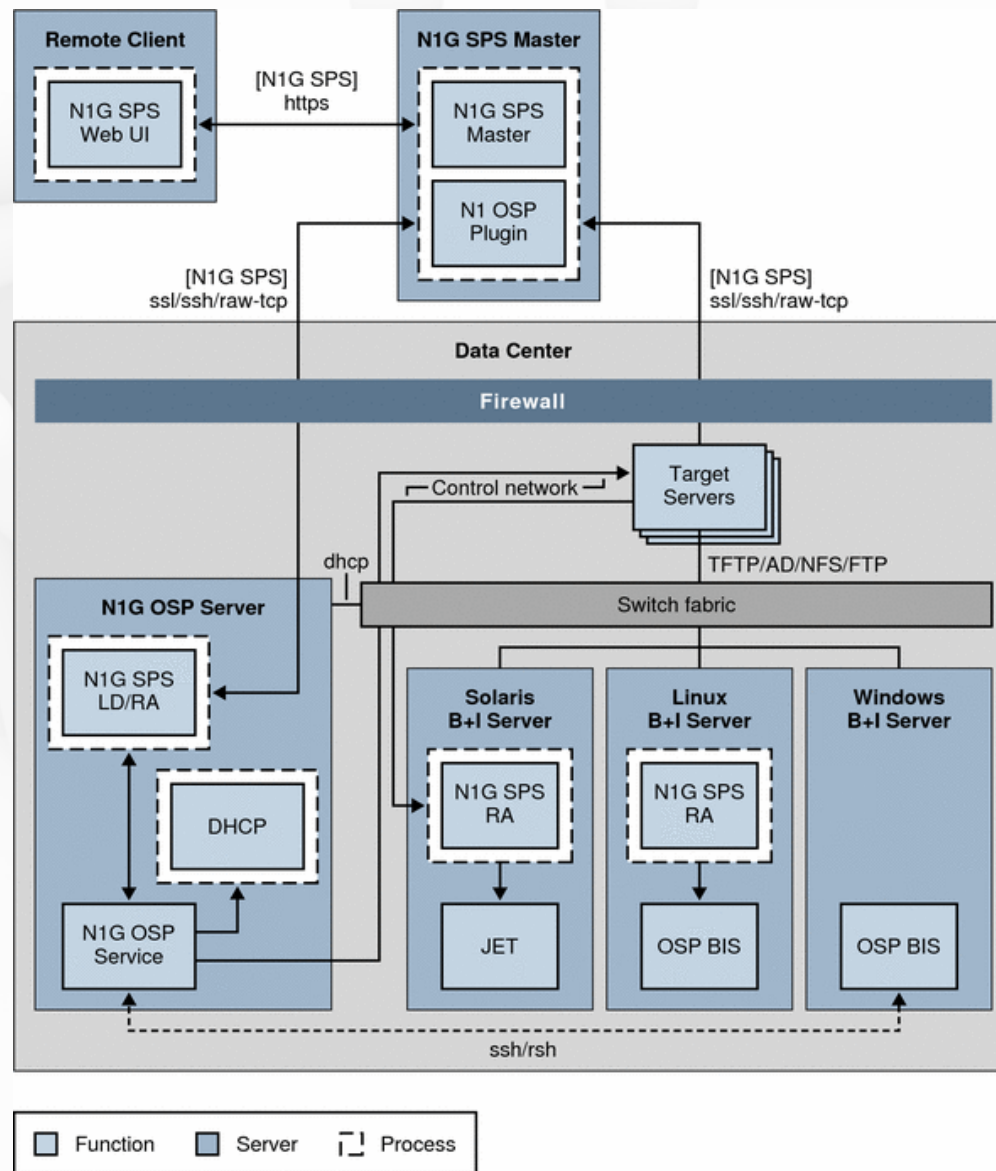


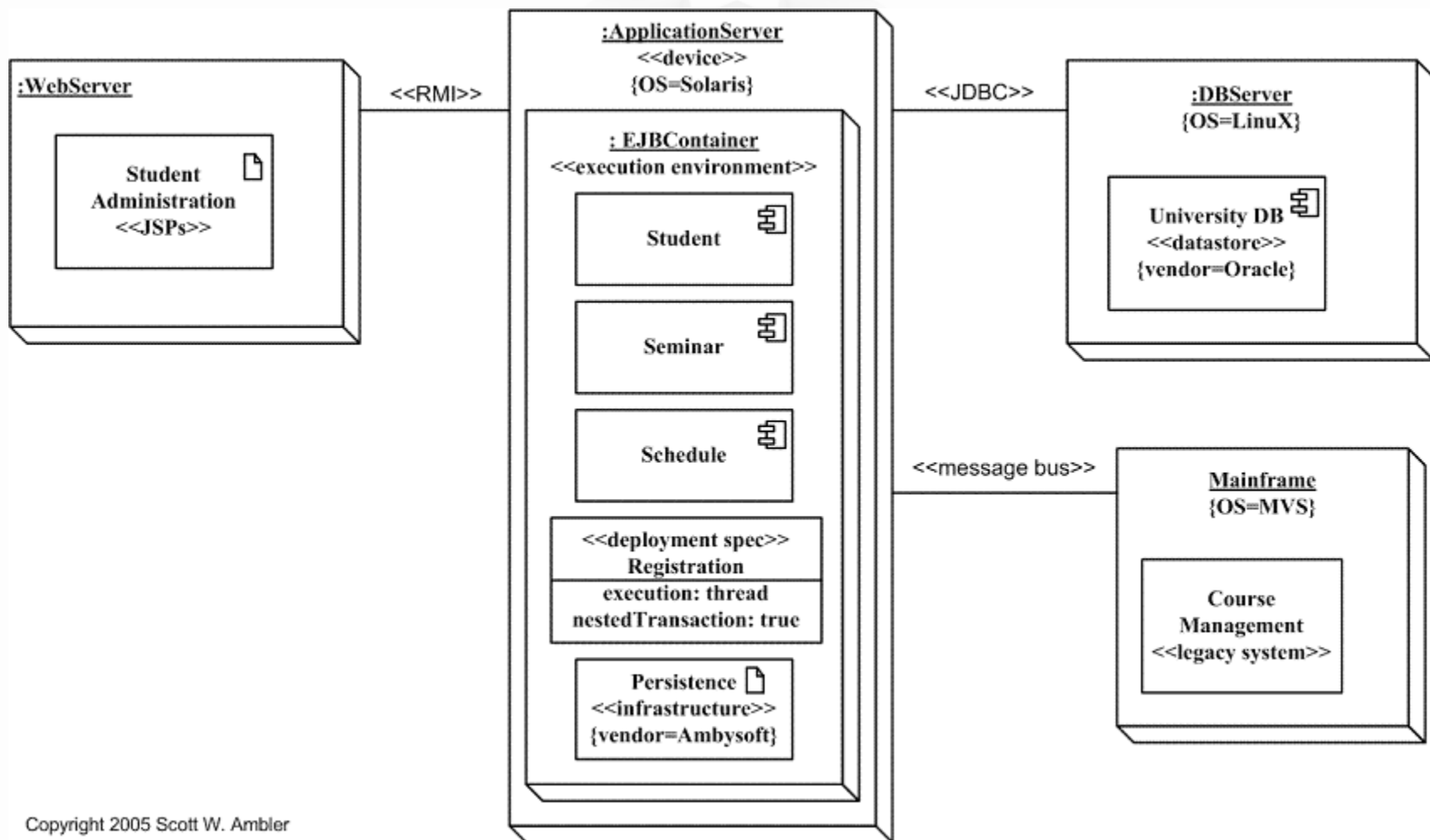












Физически изглед

