UML :

PRÉSENTATION

OBJECTIFS DU MODULE

- Concevoir des applications objets avec UML
- Décrire ce qu'est un design pattern
- Identifier les différents diagrammes
- Analyser un problème et le représenter avec UML

OBJECTIFS DU MODULE

- Formaliser les exigences sous forme de use cases
- Détailler les interactions entre objets avec les diagrammes UML
- Utiliser les dossiers de conception rédigés en UML.

PRÉREQUIS

- Idéalement : connaissance d'un langage de programmation

QU'EST-CE QUE L'UML ?

- Unified Modeling Language
- Publié en 1999 dans ***The Unified Modeling Language User Guide***
- A été mis à jour en 2005 (UML V2)

BEFORE :

- Une cinquentaine de méthodes objet (années 90)
- Pas de méthodologie commune pour un même besoin
- Existait une "guerre" des méthodologies très contre productive

MÉTHODOLOGIES DE L'ÉPOQUE

- OMT-2 : Basée sur l'analyse et l'abstraction
  - Par James Rumbaugh
- Booch'93 : Basée sur les associations, traces d'événements
  - Par Grady Booch
- OOSE : Basé sur la modélisation d'objet
  - Par Ivar Jacoson

LA NAISSANCE DE L'UML

- Mélange et amélioration des trois méthodes
- Reprenennt des idées d'autres méthodes connues

AUTRES INFLUENCES :

- Embley : Classes singletons et objets composites
- Fusion : Description des opérations, numérotation des messages
- Meyer : Pré/Post conditions
- Odell : Classification dynamique
- Shlaer-Mellor : Cycle de vie des objets

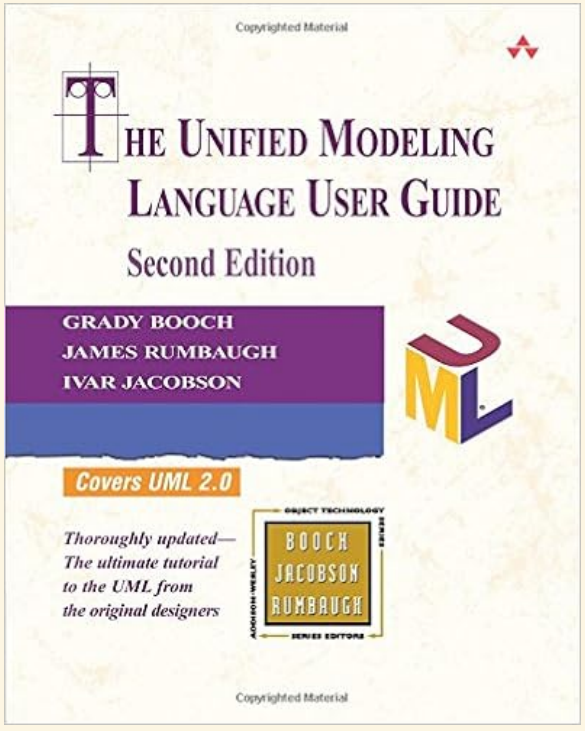
OBJECTIFS :

- Objectif : Répondre à l'augmentation de la complexité des systèmes informatiques :
  - Pour la conception,
  - Pour la compréhension informaticiens/non-informaticiens

OBJECTIFS :

- Etre capable de représenter la totalité d'un système,
- Créer un langage utilisable par les humains et les machines

THE HOLY BOOK



The Unified Modeling Language User Guide

L'UML À PROPREMENT PARLER :

- Language de modélisation pensé comme langage de modélisation commun, à la fois visible et schématique,
- Destiné à la conception de logiciels et applications,
- Permet de spécifier et documenter les artefacts des systèmes

- Se présente sous forme de plans (diagrammes),
- Dispose de ses codes (syntaxe),
- Ce n'est pas un language de programmation : il sert à concevoir et non à créer.

- Se concentre sur les artefacts du développement,
- N'a pas pour but de standardiser les processus

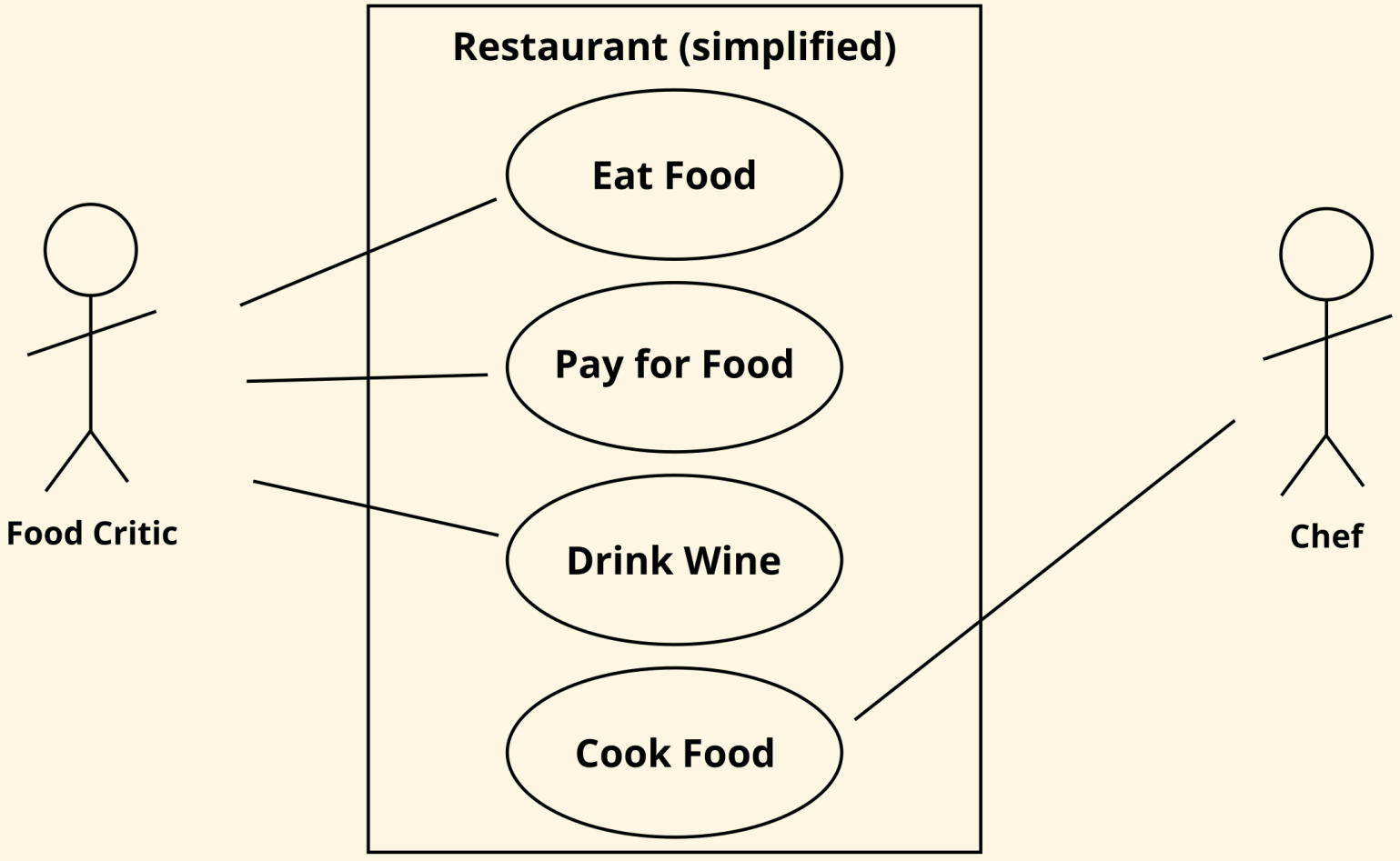
LES DIAGRAMMES

On distingue 13 types de diagrammes qui remplissent chacun des objectifs distincts. Voici la liste des plus utiles pour un développeur

DIAGRAMME DE CAS D'UTILISATION

- On y représente les différents acteurs (admins, clients, visiteurs...) et les interactions possibles entre eux et l'application
- On y tracera le cheminement des différents cas de la manière la plus exhaustive possible, en précisant ou non les prérequis des actions

EXEMPLE


Restaurant (simplified)
Eat Food
Pay for Food
Drink Wine
Cook Food
Food Critic
Chef

Finalement, une fois le diagramme d'activité fait, on pourra en tirer la liste des features de notre application !

DIAGRAMME D'ACTIVITÉ

- Représentation graphique qui représente le cycle de vie d'une activité émise par un acteur,
- Prend en compte l'aspect chronologique de l'action,
- Similaire à un workflow

EXEMPLE

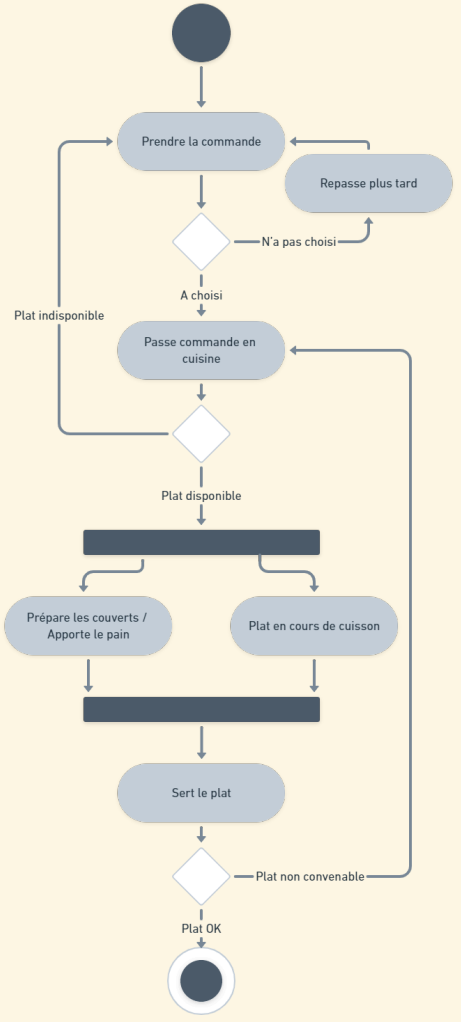

Prendre la commande
Repasse plus tard
N'a pas choisi
A choisi
Plat indisponible
Passe commande en cuisine
Plat disponible
Prépare les couverts / Apporte le pain
Plat en cours de cuisson
Sert le plat
Plat non convenable
Plat OK

DIAGRAMME DE CLASSE

- Certainement un des diagrammes les plus utilisés par les développeurs.
- Sert à décrire la structure du système

ON Y MODÉLISE :

- Les classes,
- Les attribus,
- Les opérations,
- Les relations entre objets

EXEMPLE :

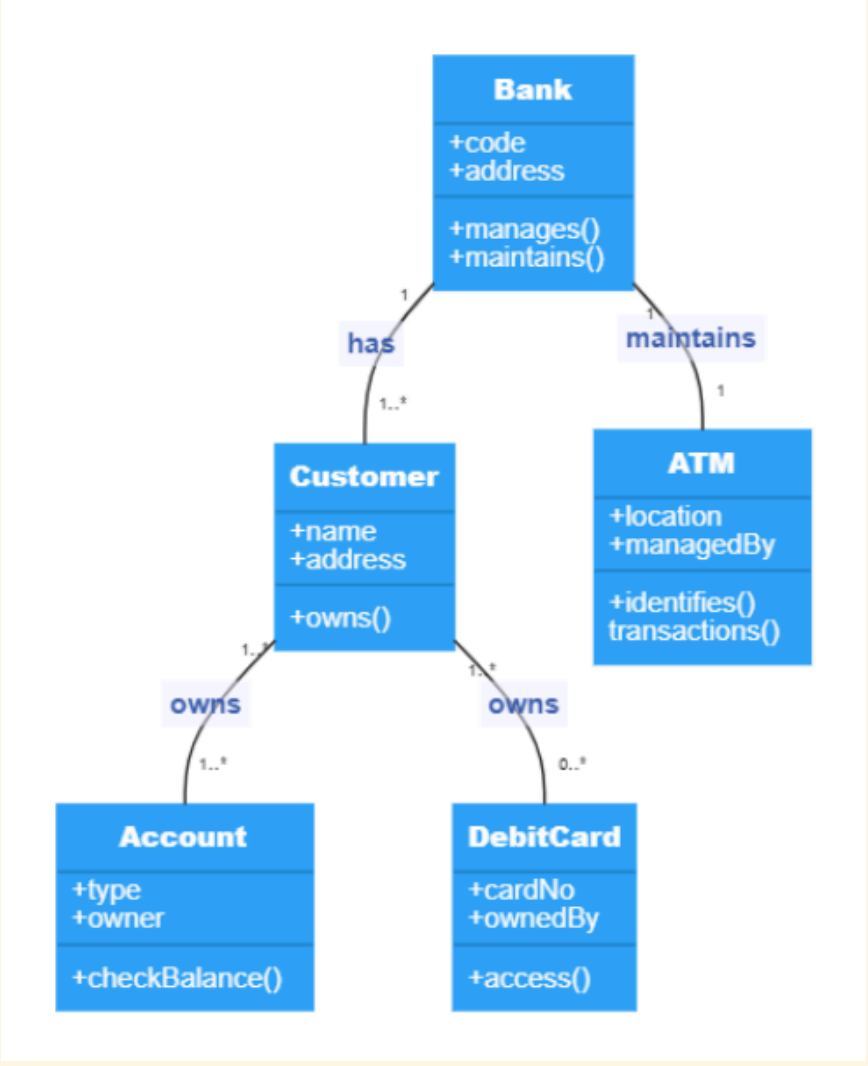

Bank
+code
+address
+manages()
+maintains()
1
has
1..*
1
maintains
1
Customer
+name
+address
+owns()
ATM
+location
+managedBy
+identifies()
transactions()
1..*
owns
1..*
1..*
owns
0..*
Account
+type
+owner
+checkBalance()
DebitCard
+cardNo
+ownedBy
+access()

DIAGRAMME D'OBJET

Représente une instance spécifique d'un diagramme à un moment précis. Il ressemble beaucoup au diagramme de classe, si bien qu'il sont souvent mixés.

EXEMPLE

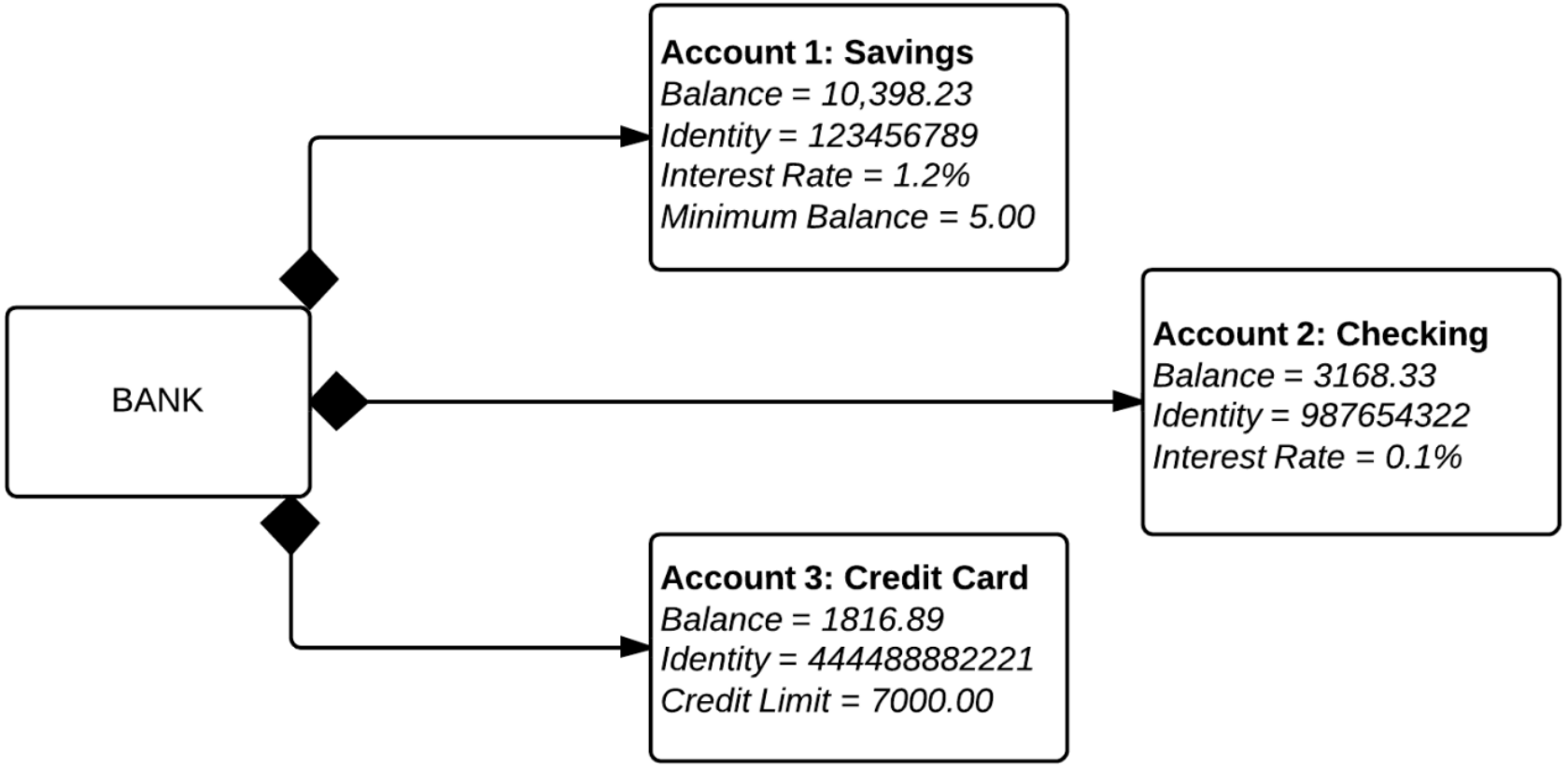
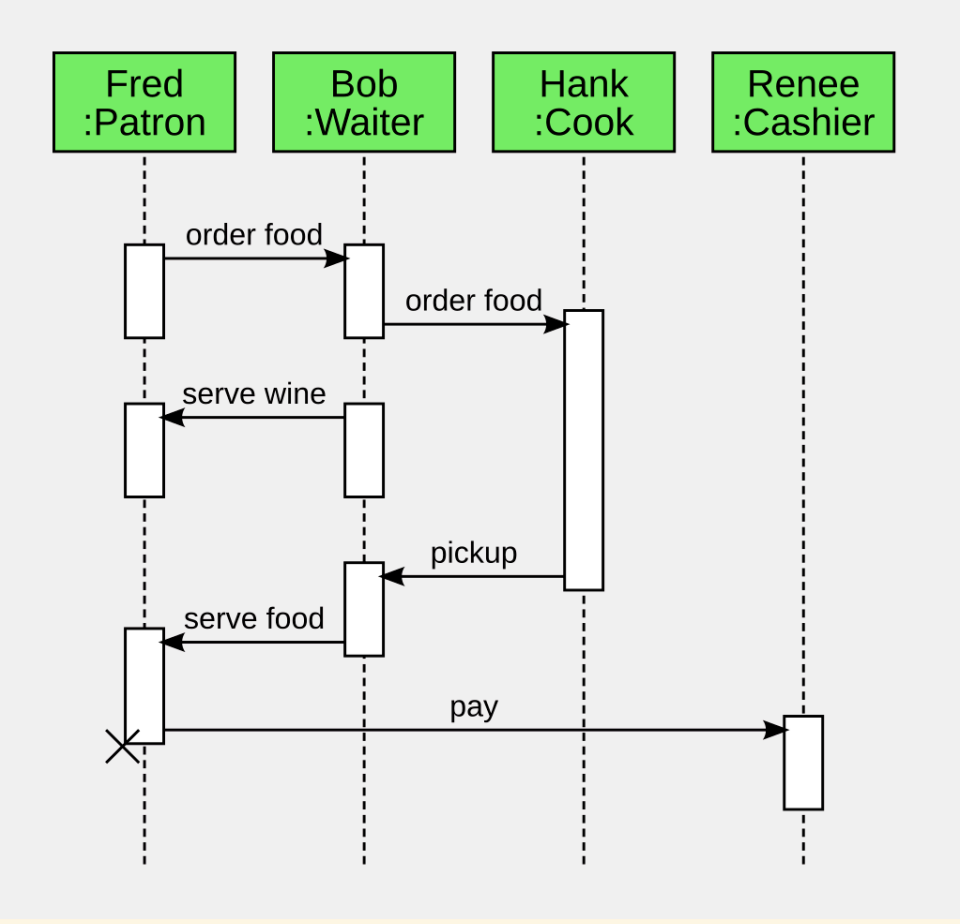

BANK
Account 1: Savings
Balance = 10,398.23
Identity = 123456789
Interest Rate = 1.2%
Minimum Balance = 5.00
Account 2: Checking
Balance = 3168.33
Identity = 987654322
Interest Rate = 0.1%
Account 3: Credit Card
Balance = 1816.89
Identity = 444488882221
Credit Limit = 7000.00

DIAGRAMME DE SÉQUENCE

- Sert à représenter les interactions entre acteurs,
- Prend en compte la gestion du temps
- Peuvent être appelés "Diagrammes d'évènements" ou "Scénario d'évènements"

EXEMPLE


Fred
:Patron
Bob
:Waiter
Hank
:Cook
Renee
:Cashier
order food
order food
serve wine
pickup
serve food
pay

AUTRES DIAGRAMMES

Voici les autres diagrammes UML existants qui ne seront pas abordés en détail durant ce cours.

DIAGRAMME DE COMPOSANT

- Permet d'identifier les composants du système
  - Structure physique,
  - Relations entre composants,
  - Comportement services/interface

EXEMPLE

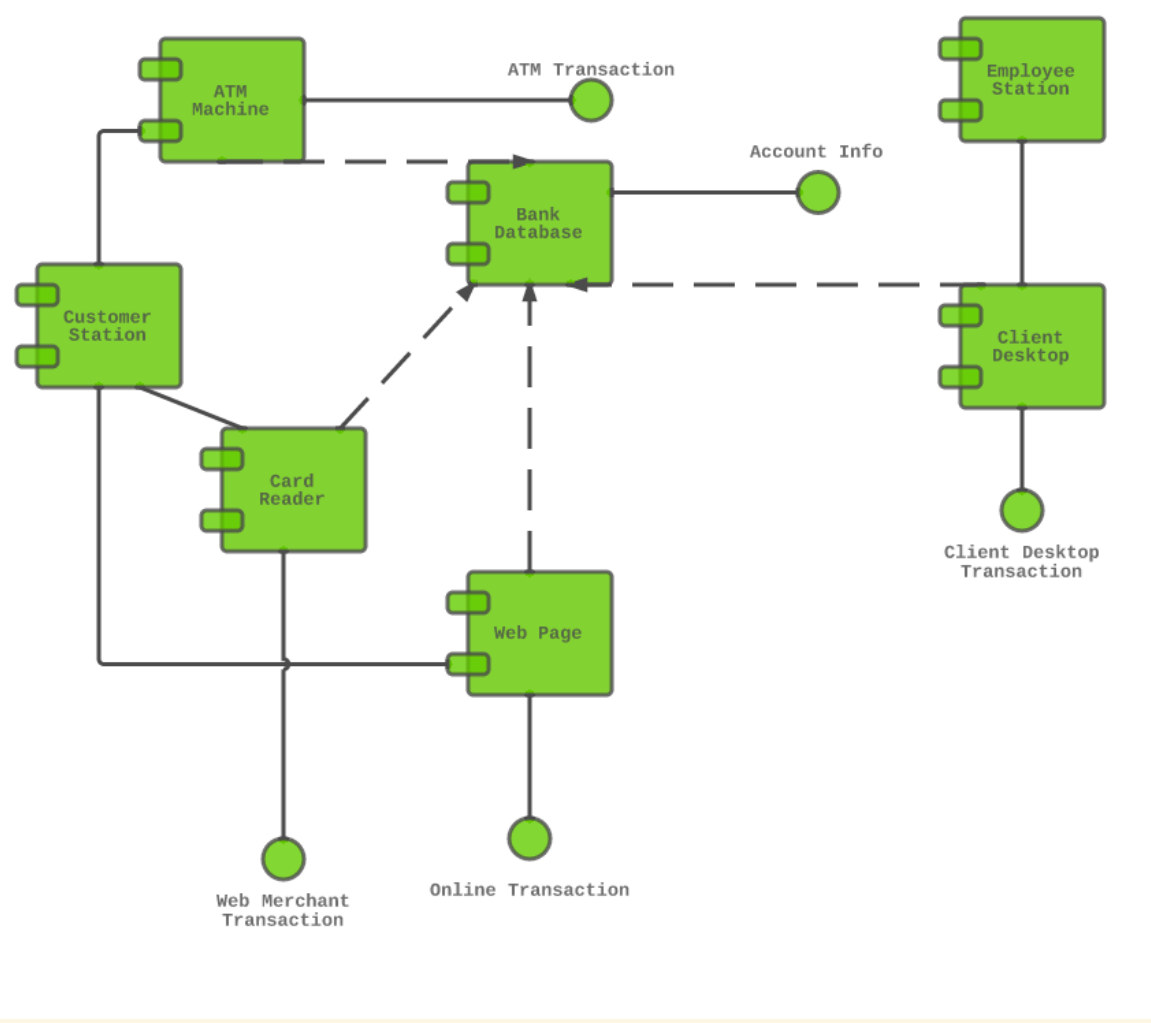

ATM Machine
ATM Transaction
Employee Station
Account Info
Bank Database
Customer Station
Client Desktop
Card Reader
Client Desktop Transaction
Web Page
Web Merchant Transaction
Online Transaction

DIAGRAMME DE DÉPLOIEMENT

- Permet de faire le lien entre les composants logiciels et matériels,
- Illustre le traitement/l'exécution
- Permet de visualiser les composants matériels (serveurs, etc.)

EXEMPLE

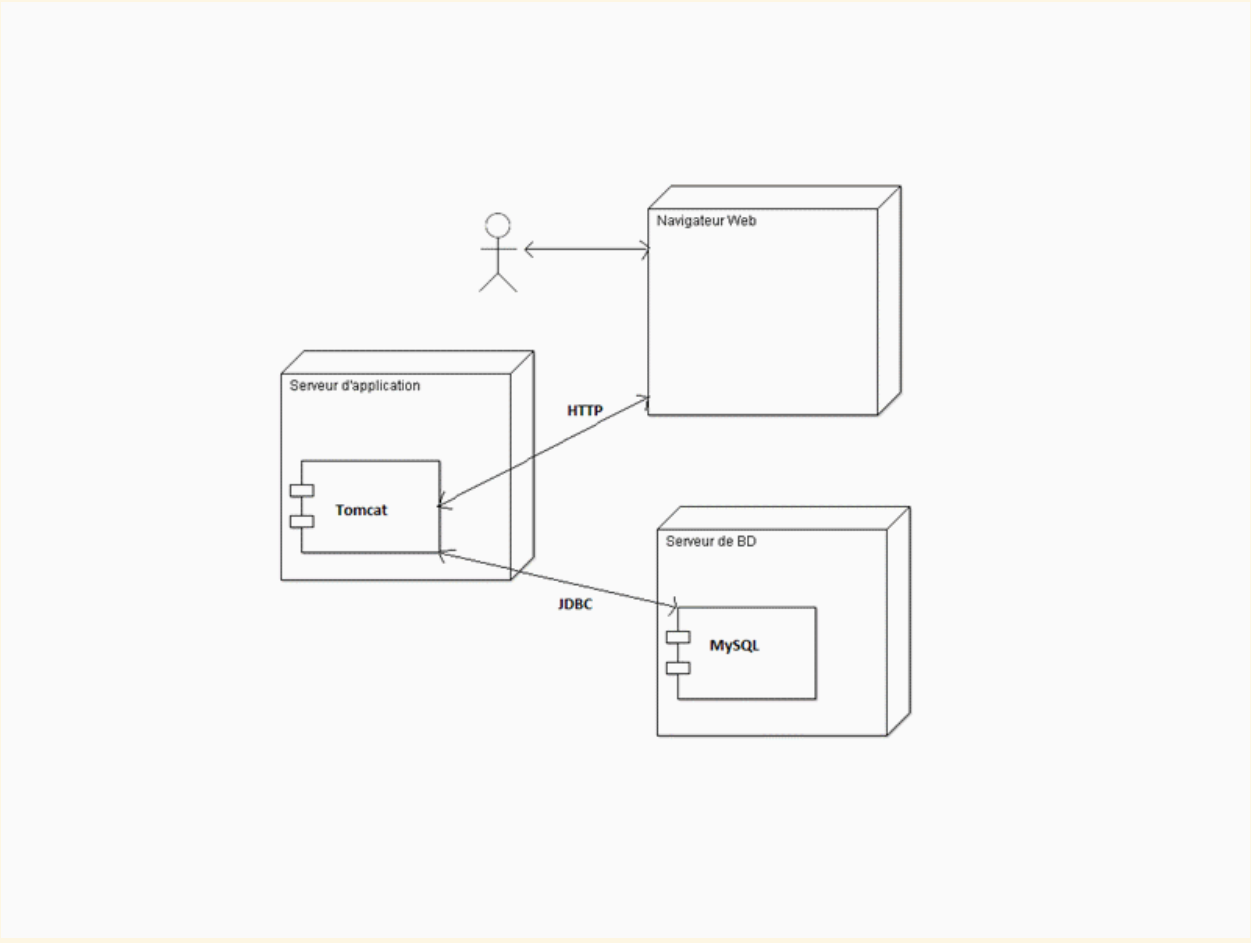
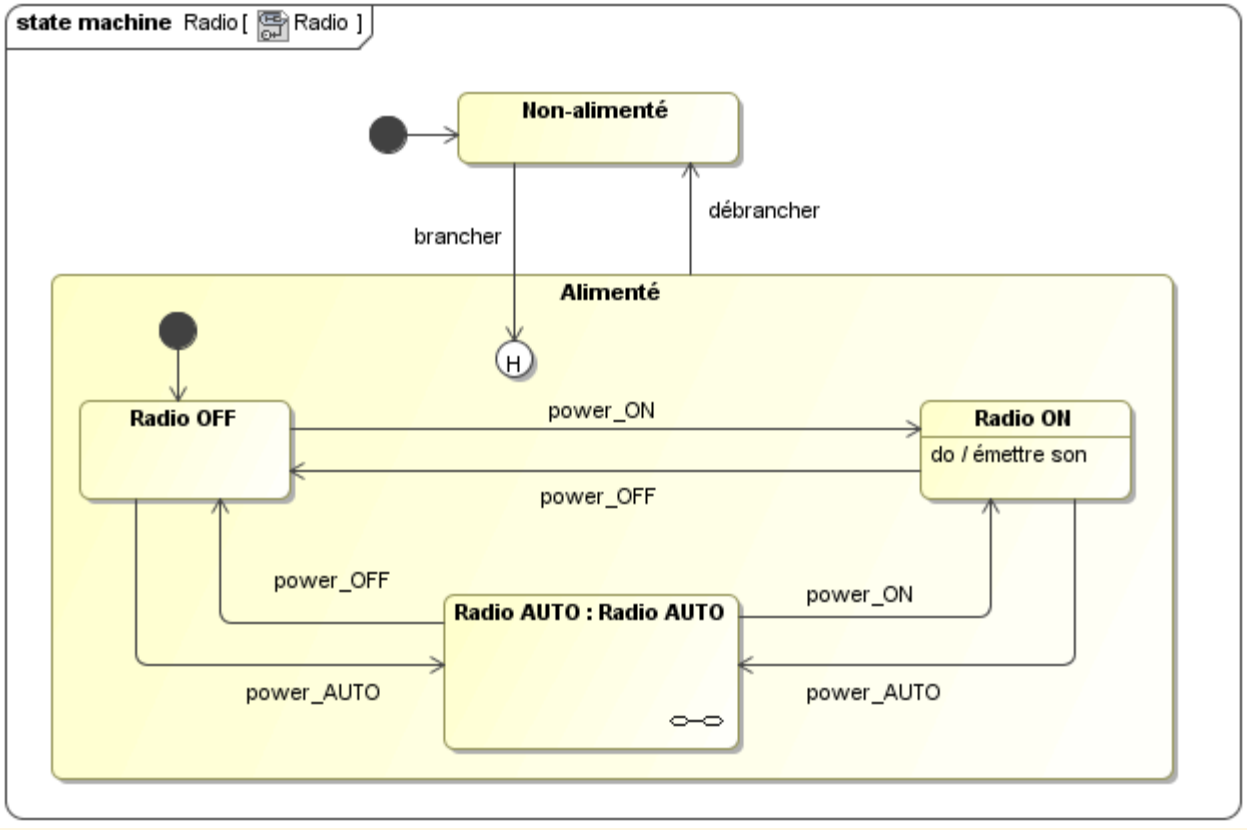

Navigateur Web
Serveur d'application
HTTP
Tomcat
Serveur de BD
JDBC
MySQL

DIAGRAMME D'ÉTAT

- Se concentre sur un objet/ensemble d'objet,
- Fait état des différents cycles d'évolution possible
- Ressemble beaucoup au diagramme d'activité, si bien que les deux sont souvent mélangés

EXEMPLE


state machine Radio [Radio]
Non-alimenté
brancher
débrancher
Alimenté
H
Radio OFF
power_ON
Radio ON
do / émettre son
power_OFF
power_OFF
power_ON
Radio AUTO : Radio AUTO
power_AUTO
power_AUTO

LES OUTILS DE RÉDACTION UML

Dans ce cours, nous utiliserons PlantText qui permet de traduire une notation en schéma. La documentation peut être trouvée sur ce site

PLANTEXT

- On utilisera des symboles pour illustrer les éléments que l'on souhaite afficher
- Plantext interprêtera les informations pour créer le diagramme en question

EXEMPLE

```
@startuml
left to right direction
actor "Food Critic" as fc
rectangle Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
@enduml
```