Conception Orientée Objet & UML

Présentation du module

Licence Professionnelle IDSE
2015-2016

IUT de Nice

Simon Urli
urli@i3s.unice.fr

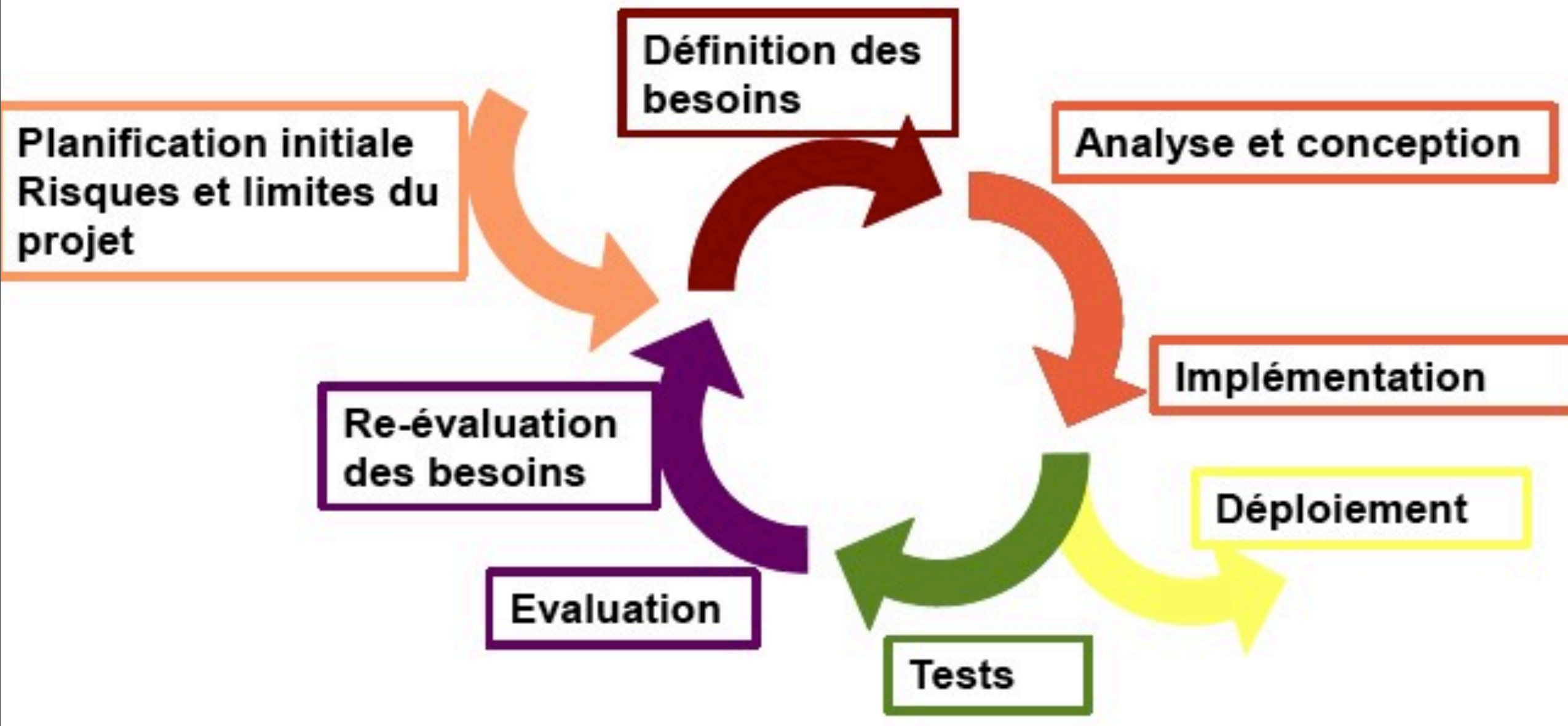
Inspiré du cours de Mireille Blay-Fornarino

Selon vous à quoi sert ce module ?

Introduction à la conception

Vous devez réaliser une solution logicielle pour visualiser les données météo dans un cockpit, que faites vous ?

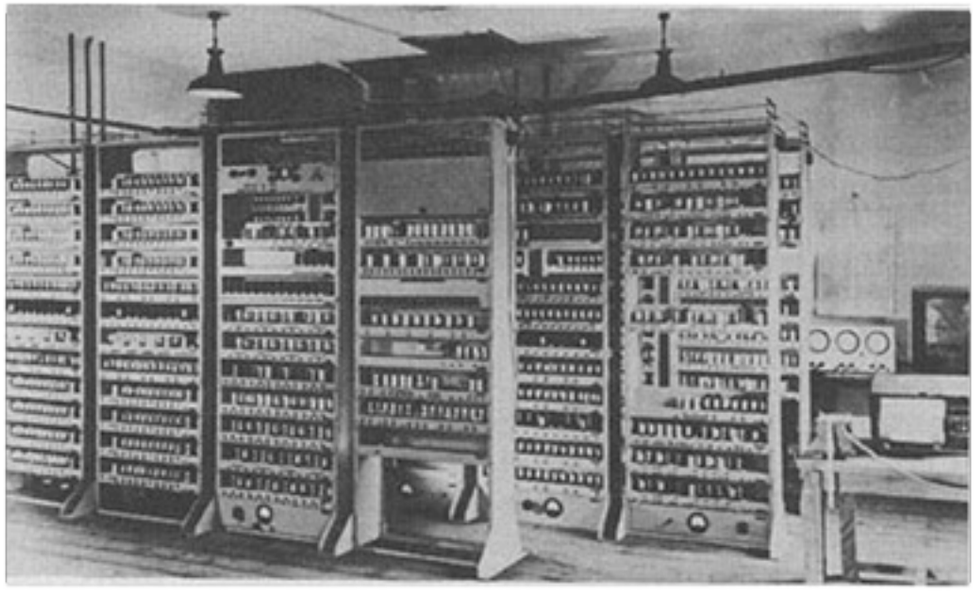
Activités du développement logiciel



Histoire de la gestion de la complexité dans le développement logiciel

Premiers programmes : langage machine

- Dépendants de l'architecture des ordinateurs
- Un jeu d'instruction = un type de machine
- Portage => réécriture du programme



```
sub     $sp, $sp, 4
sw      r,0($sp)
```

Création des premiers langages structurés

- Structures de contrôles (if, while, do, goto...)
- Séparer les concepts du langage de leur représentation en langage machine
- Pouvoir écrire un seul programme pour plusieurs architectures distinctes

```
      PROGRAM DEGRAD
!
! Imprime une table de conversion degrés -> radians
! ================================================
!
! Déclaration des variables
      INTEGER DEG
      REAL RAD, COEFF
!
! En-tête de programme
      WRITE ( *, 10)
   10 FORMAT      (' ',20('*') /              &
     &             ' * Degres * Radians *' /  &
     &             ' ', 20('*') )
!
! Corps de programme
      COEFF = (2.0 * 3.1416) / 360.0
      DO DEG = 0, 90
         RAD = DEG * COEFF
         WRITE ( *, 20) DEG, RAD
   20 FORMAT         (' *  ',I4,'  * ',F7.5,' *')
      END DO
!
! Fin du tableau
      WRITE ( *, 30)
   30 FORMAT      (' ',20('*') )
!
! Fin de programme
      STOP
      END PROGRAM DEGRAD
```

Développement par décomposition

- Travail sur la structure des programmes pour éviter la programmation spaghetti
- Evolution des langages pour pouvoir définir des modules fonctionnelles dans les programmes
- Stratégie de type «diviser pour régner»

Explosion des besoins

- Explosion de la miniaturisation électronique
- Chûte des coûts du matériel informatique
- L'informatique est devenu courant à la fois en entreprise et chez les particuliers
- Explosion des besoins logiciels dans tous les domaines
- Comment créer plus vite et mieux plus de logiciels plus complexes ?

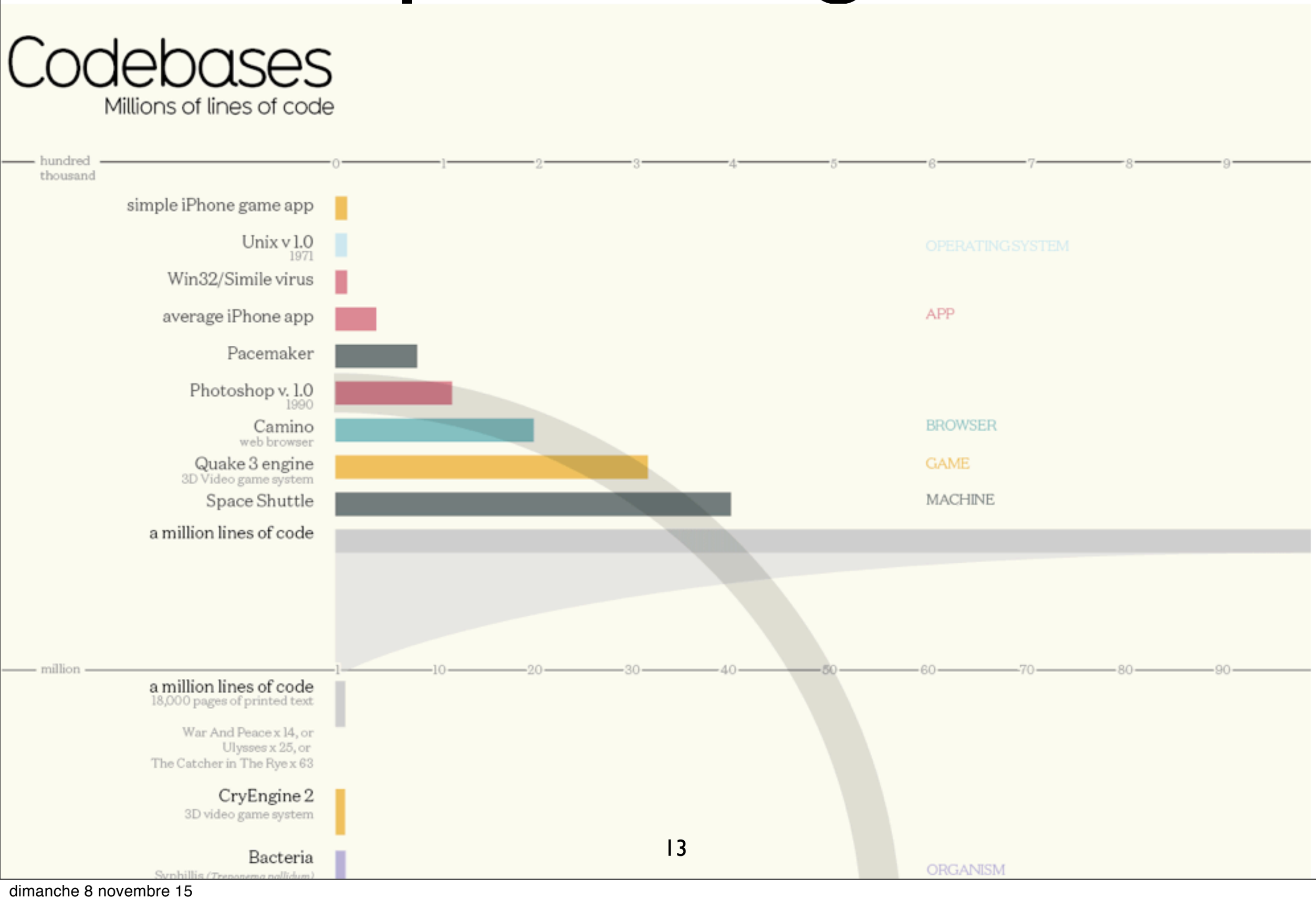
Avènement du paradigme objet

- Montée en abstraction : évolution des langages pour pouvoir définir la notion d'objets
- Définition des notions de contrats, de composants, de services, de frameworks
- Automatisation via la génération de code à partir d'abstractions
- Ingénierie des modèles

Et après ???

- Recherche en génie logiciel
- Vers des usines de logiciels ?
- Travail sur des logiciels qui réutilisent des logiciels ?
- La difficulté de la gestion de l'évolution !
- Abstraire plus ? Concilier abstraction et spécification du domaine ?

Complexité logicielle ?



v10.4

0

1 10 20 30 40 50 60 70 80 90

Car software
average modern high-end car

Mouse*
Total DNA basepairs in genome

billion

1 10 20 30 40 50 60 70 80 90

Google
all internet services

Echecs logiciels

- 1985-87 : Therac 25 - Appareil de radiothéraphie. Bug dans le logiciel d'adaptation des radiations => Plusieurs morts
- 1990 : Crash du réseau AT&T pendant 9 heures : une seule ligne de code en faute
- 1996 : Interruption de la mission Ariane 5 : Problème d'encodage des entiers.
- Et d'autres !

Diviser pour résoudre

- Techniquement en s'appuyant sur la modularité :
  - Encapsulation, faible couplage
  - Notion d'objets
- En s'organisant au delà du seul développement :
  - Analyse et conception
  - Validation et vérification

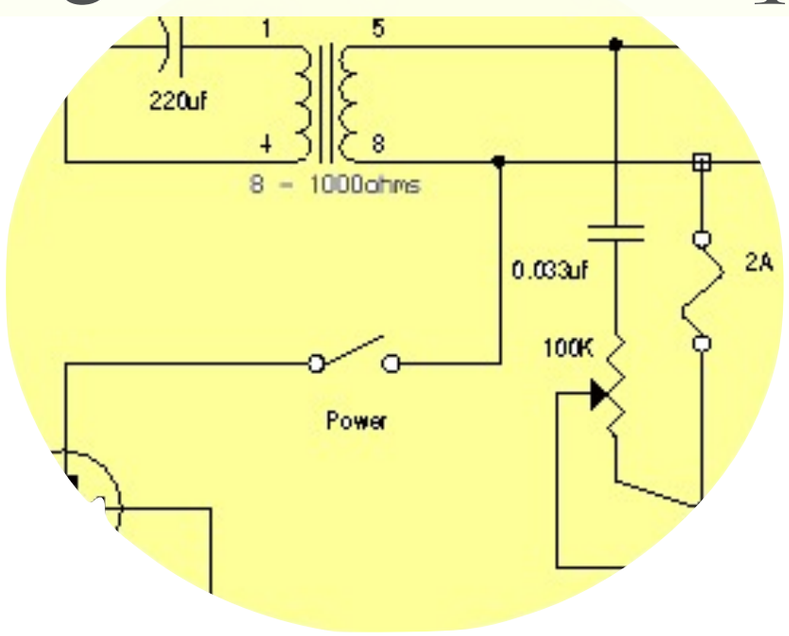
Développer dans la durée grâce à la modélisation

- Techniquement :
  - Assurer une meilleure continuité entre le «domaine du problème» et le «domaine des solutions»
  - Définir les fonctionnalités de manière à prévoir les tests et à considérer tout de suite la maintenance
- Du point de vue de l'organisation :
  - Tracer les changements
  - Gérer l'évolution
  - Utiliser des méthodes agiles

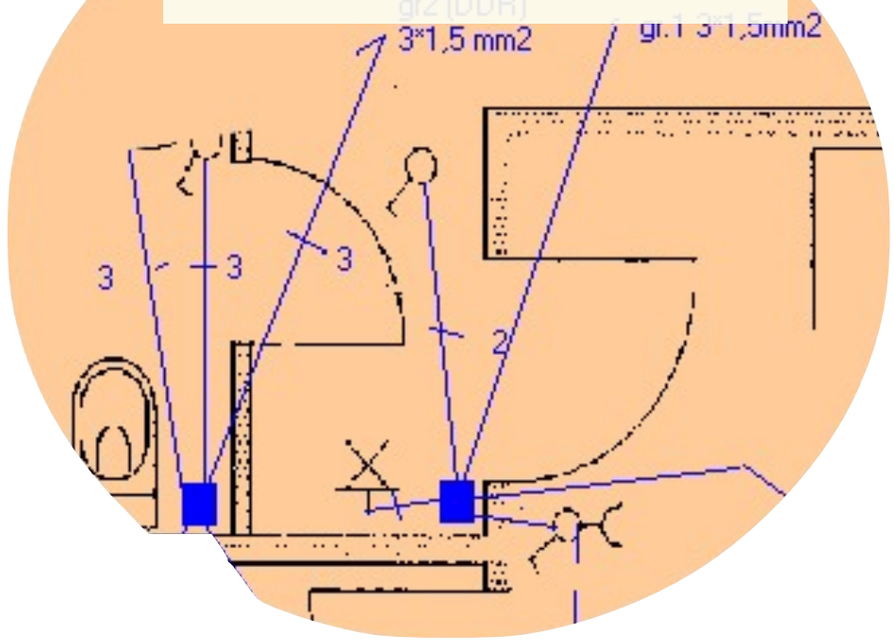
UNIFIED
MODELING
LANGUAGE

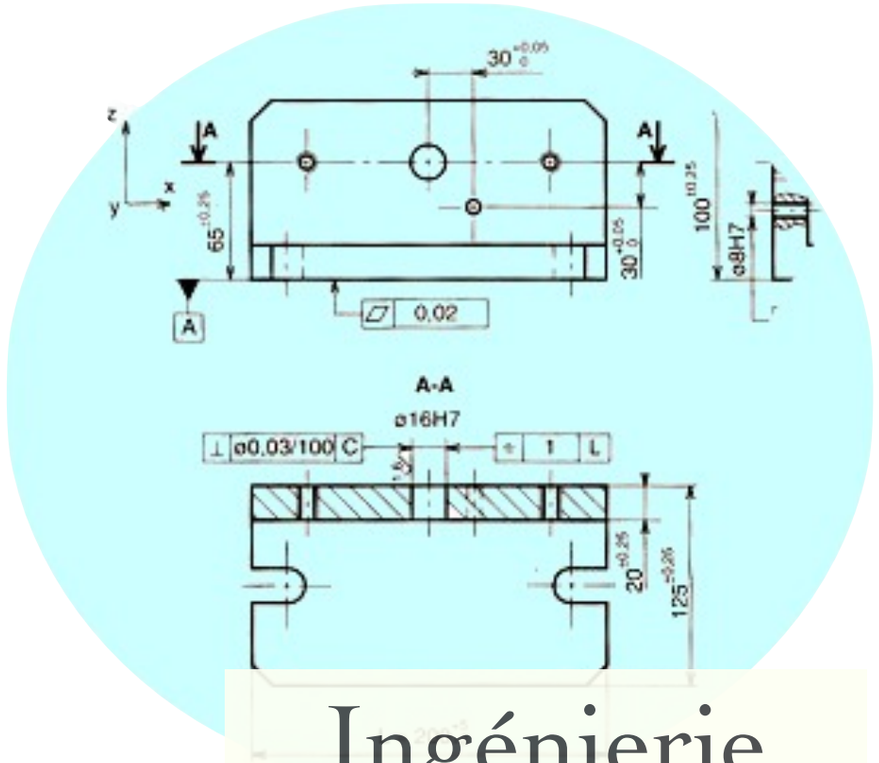
Introduction à UML

Ingénierie Électrique

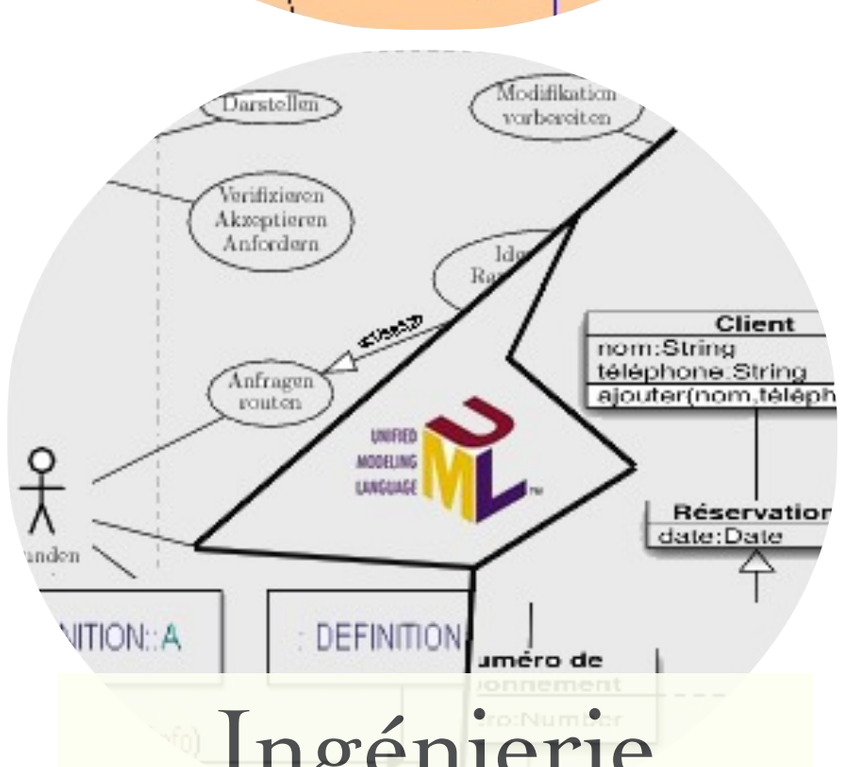


Ingénierie du Bâtiment





Ingénierie Mécanique



Ingénierie Logicielle

UML : un support à la modélisation

- Modèle : simplification de la réalité dans le but de :

UML : un support à la modélisation

- Modèle : simplification de la réalité dans le but de :

Visualiser

le système

UML : un support à la modélisation

- Modèle : simplification de la réalité dans le but de :

Visualiser le système

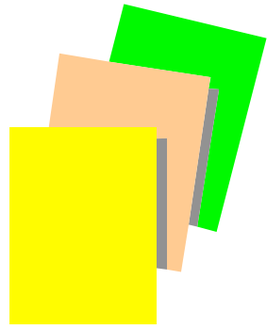
Spécifier la **structure et le comportement** du système

UML : un support à la modélisation

- Modèle : simplification de la réalité dans le but de :

Visualiser le système

Spécifier la **structure et le comportement** du système



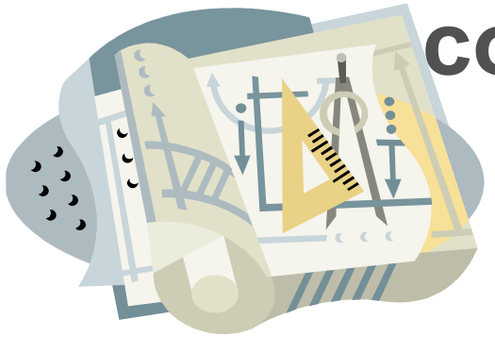
Documenter ***les décisions***

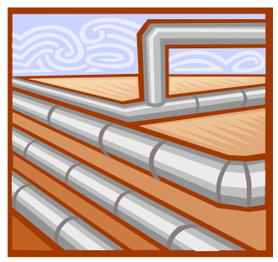
UML : un support à la modélisation

- Modèle : simplification de la réalité dans le but de :

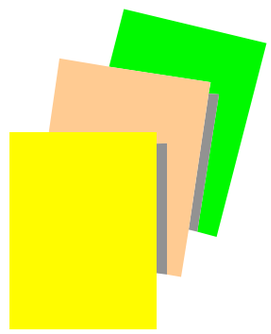
Visualiser le système

Spécifier la **structure et le comportement** du système





Aider à la construction du système



Documenter les décisions

Qu'est-ce qu'UML ?

- UML est un **langage visuel**
- Il supporte :
  - la visualisation
  - la spécification
  - la construction
  - la documentation

- Descriptions graphiques et textuelles
- Syntaxe et sémantique
- Architecture et comportement
- Génération de code

Points forts d'UML

- UML est un langage **normalisé**
  - Gain de précision
  - Gage de stabilité
  - Encourage l'utilisation d'outils
- UML est un support de communication performant
  - Permet de cadrer l'analyse
  - Facilite la compréhension de systèmes complexes
  - Langage universel en ingénierie logiciel

Points forts d'UML

- UML est un langage **normalisé**
  - Gain de précision
  - Gage de stabilité
  - Encourage l'utilisation d'outils
- UML est un support de communication performant
  - Permet de cadrer l'analyse
  - Facilite la compréhension de systèmes complexes
  - Langage universel en ingénierie logiciel

Il doit vous aider à comprendre les concepts de la programmation par objets!

Points forts d'UML

- UML est un langage **normalisé**
  - Gain de précision
  - Gage de stabilité
  - Encourage l'utilisation d'outils
- UML est un support de communication performant
  - Permet de cadrer l'analyse
  - Facilite la compréhension de systèmes complexes
  - Langage universel en ingénierie logiciel

absence de rigueur

Il doit vous aider à comprendre les concepts de la programmation par objets!

Points faibles d'UML

- Nécessite une période d'apprentissage et d'adaptation
- Beaucoup de concepts dans UML
- Ne suffit pas en soi à réussir un projet : le processus de développement, la qualité du code, etc sont nécessaires
- Toujours pas la panacée universelle !

Introduction à UML

survol

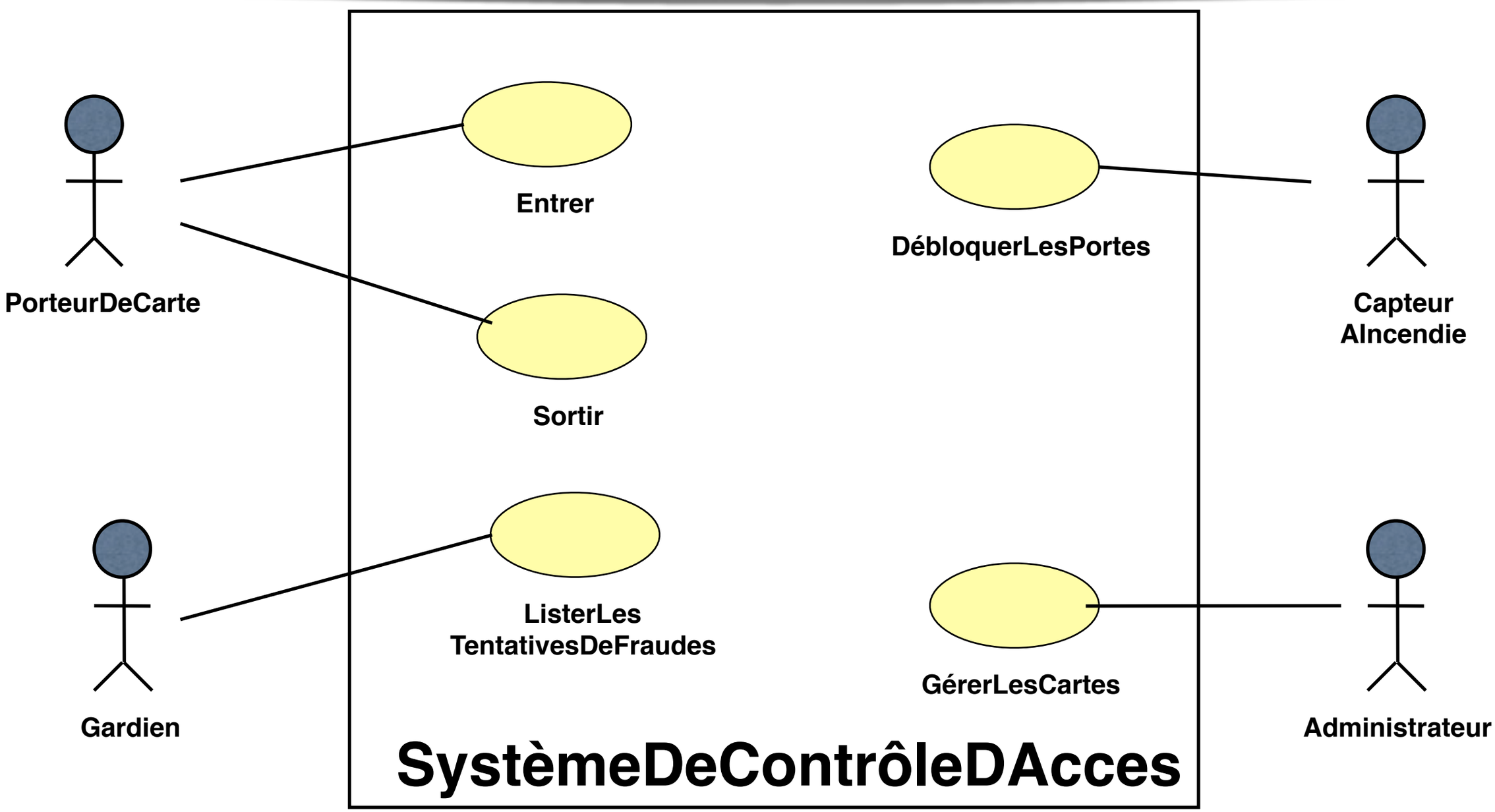
Vue fonctionnelle

la vue fonctionnelle cherche à appréhender
les interactions entre les
acteurs/utilisateurs
et le système,
sous forme d'objectifs à atteindre (**cas d'utilisation**) et sous forme chronologique de scénarios d'interaction typiques (**diagrammes de séquences**)

http://wwwdfr.ensta.fr/Cours/docs/IN204/uml.pdf

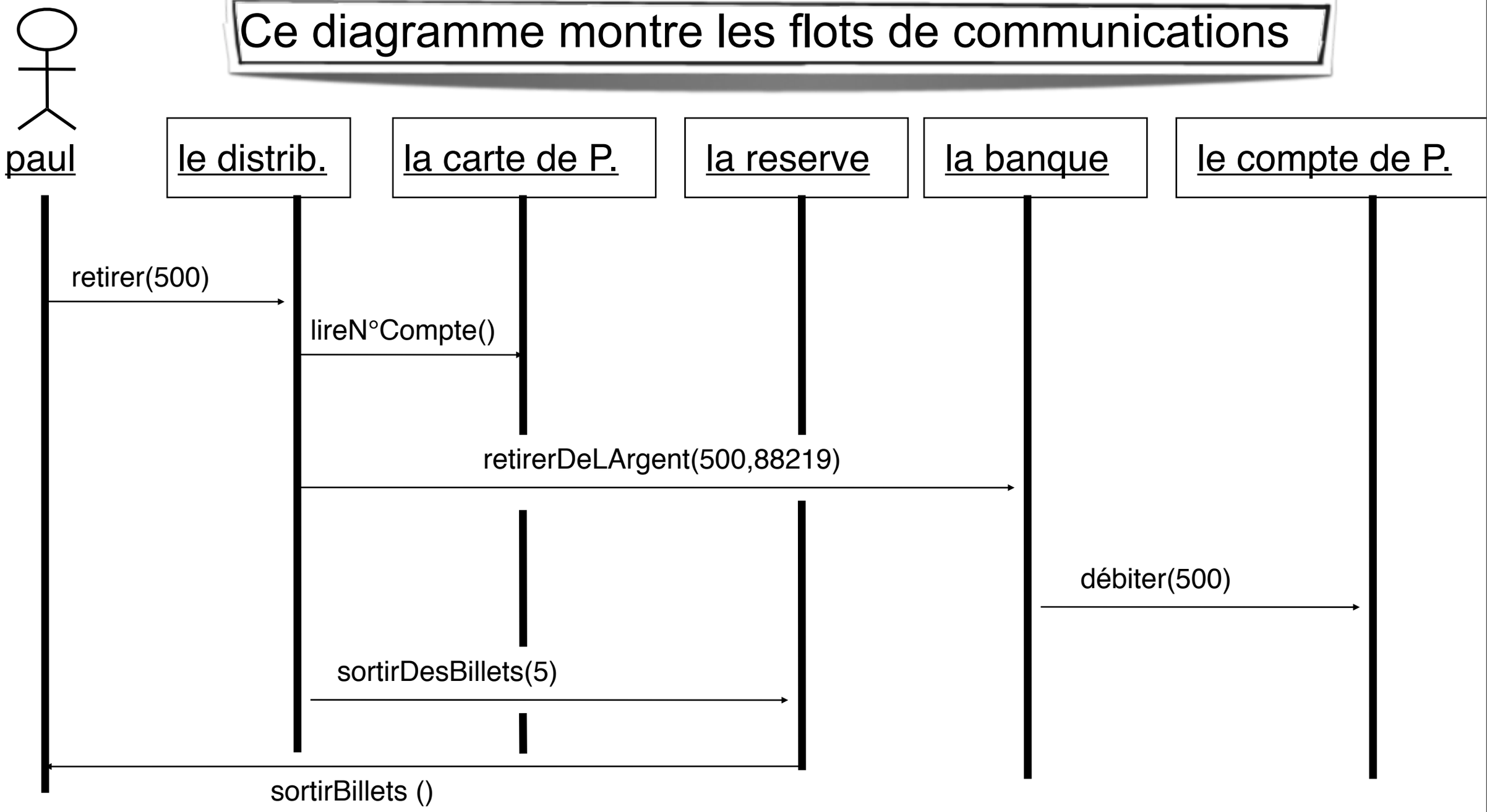
Diagrammes des cas d'utilisation

Ce diagramme montre ce que que fait le système et qui l'utilise



Diagrammes de séquence

Ce diagramme montre les flots de communications

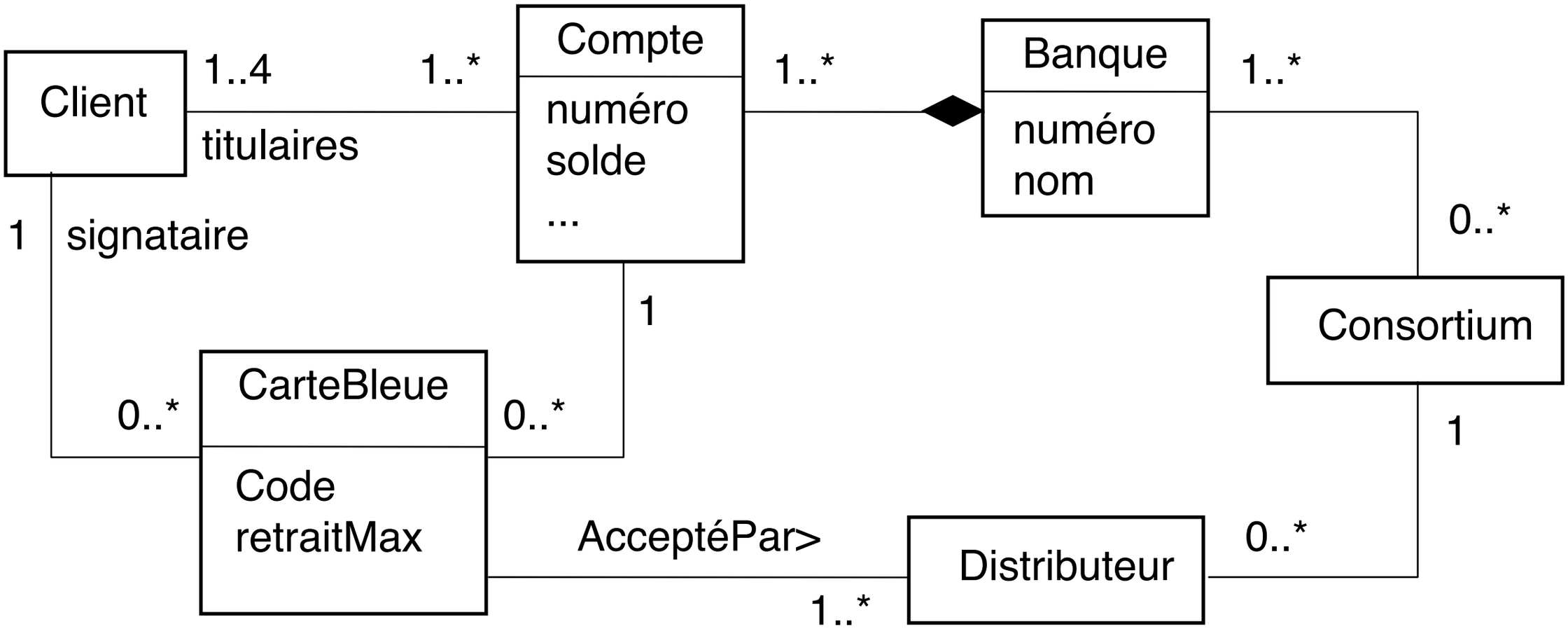


Vue Structurelle

la vue structurelle, ou statique, vise à **identifier les objets/composants** constituant le programme, leurs attributs, opérations et méthodes, ainsi que les liens ou associations qui les unissent (**diagramme de classes**). Elle permet aussi de regrouper les classes fortement liées entre elles en des composants les plus autonomes possibles (**diagramme de packages**).
A l'intérieur de chaque package, on trouve un diagramme de classes.

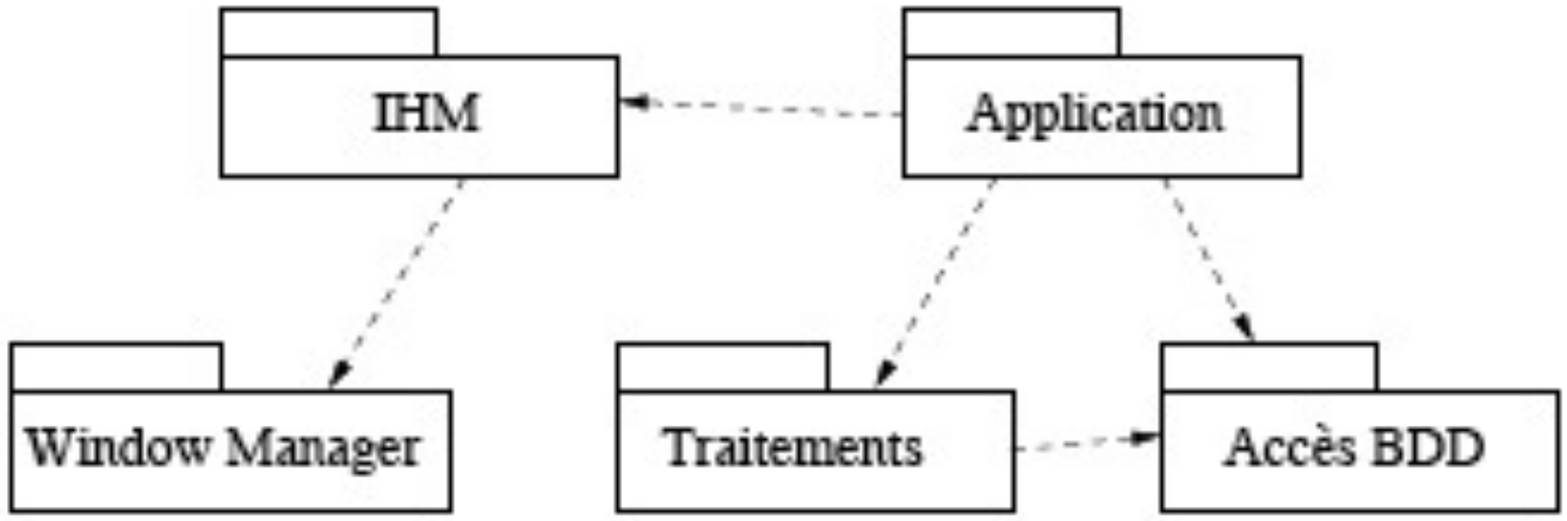
http://wwwdfr.ensta.fr/Cours/docs/IN204/uml.pdf

Diagrammes de classes



Ce diagramme montre les classes et les relations entre elles

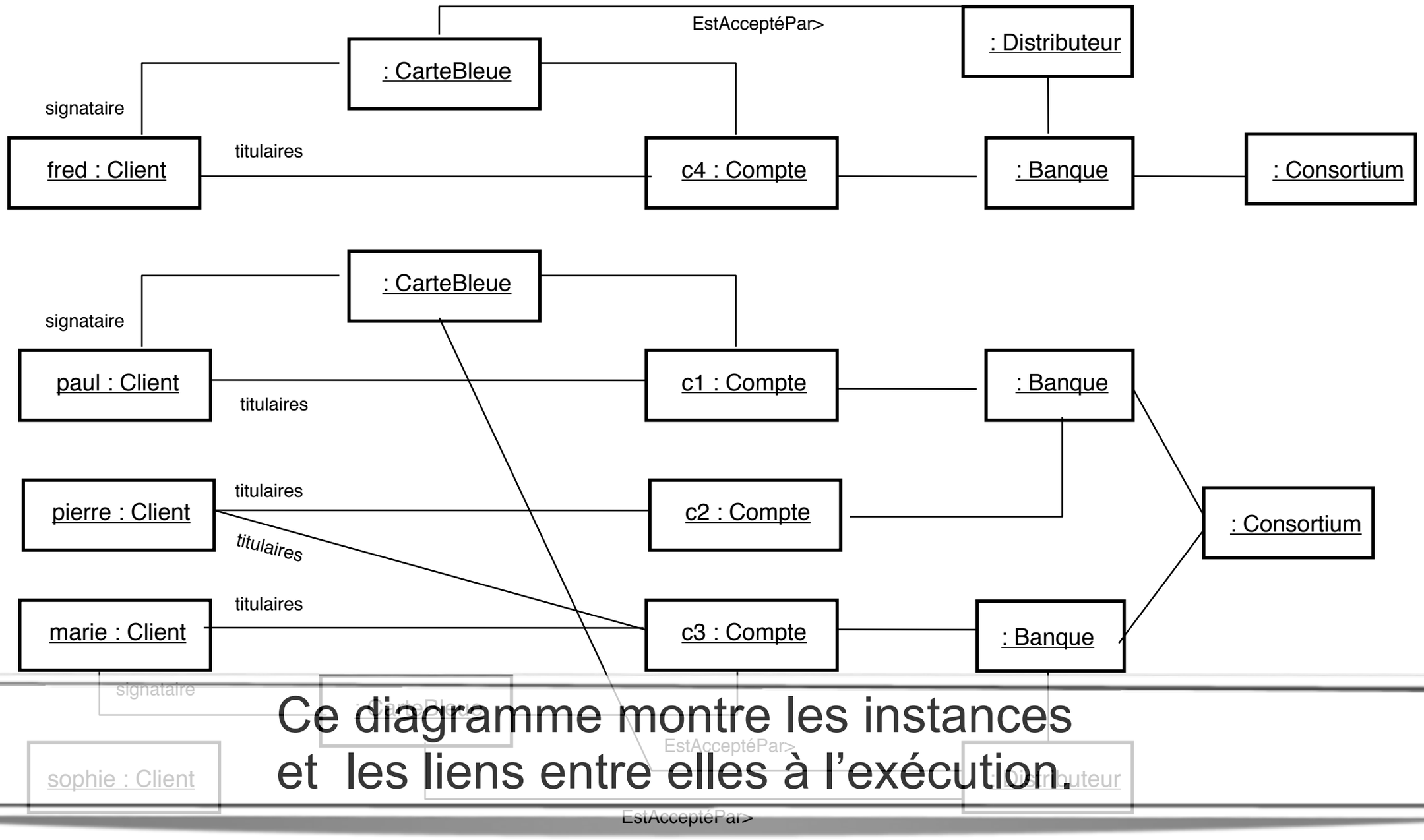
Diagrammes de packages



Regrouper entre elles des classes liées les unes aux autres de manière à faciliter la maintenance ou l'évolution du projet et de rendre aussi indépendantes que possible les différentes parties d'un logiciel.

http://wwwdfr.ensta.fr/Cours/docs/IN204/uml.pdf

Diagrammes d'objets



Ce diagramme montre les instances et les liens entre elles à l'exécution.

Vue Dynamique

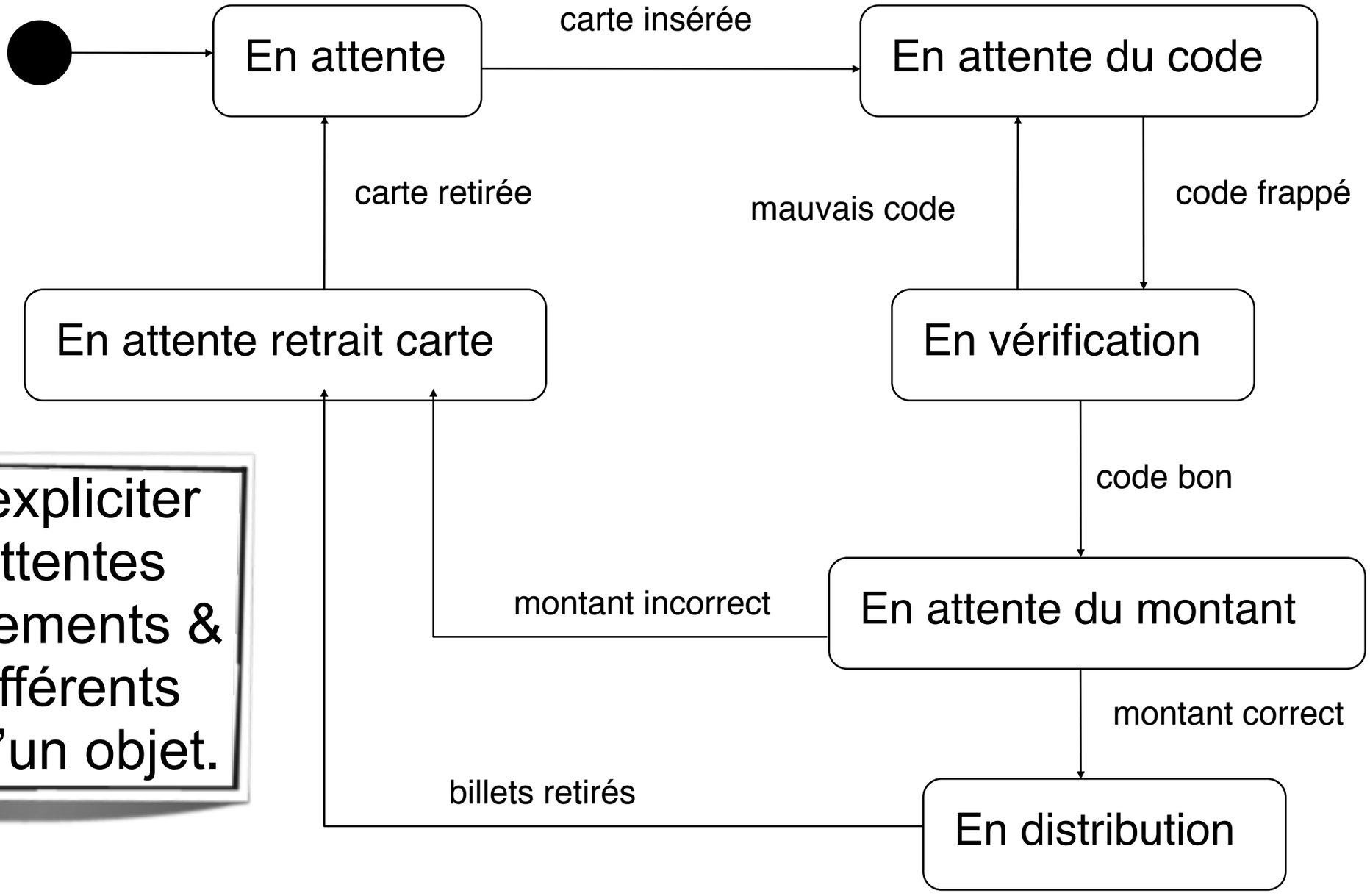
la vue dynamique vise à décrire
l'évolution (la dynamique) des objets complexes du programme tout au long de leur cycle de vie.

De leur naissance à leur mort, les objets voient leurs changement d'états guidés par les interactions avec les autres objets(les **diagrammes d'états**).

Le **diagramme d'activité** est une sorte d'organigramme correspondant à une version simplifiée du diagramme d'états. Il permet de modéliser des activités qui se déroulent en parallèle les unes des autres, quand ce parallélisme peut poser problème.

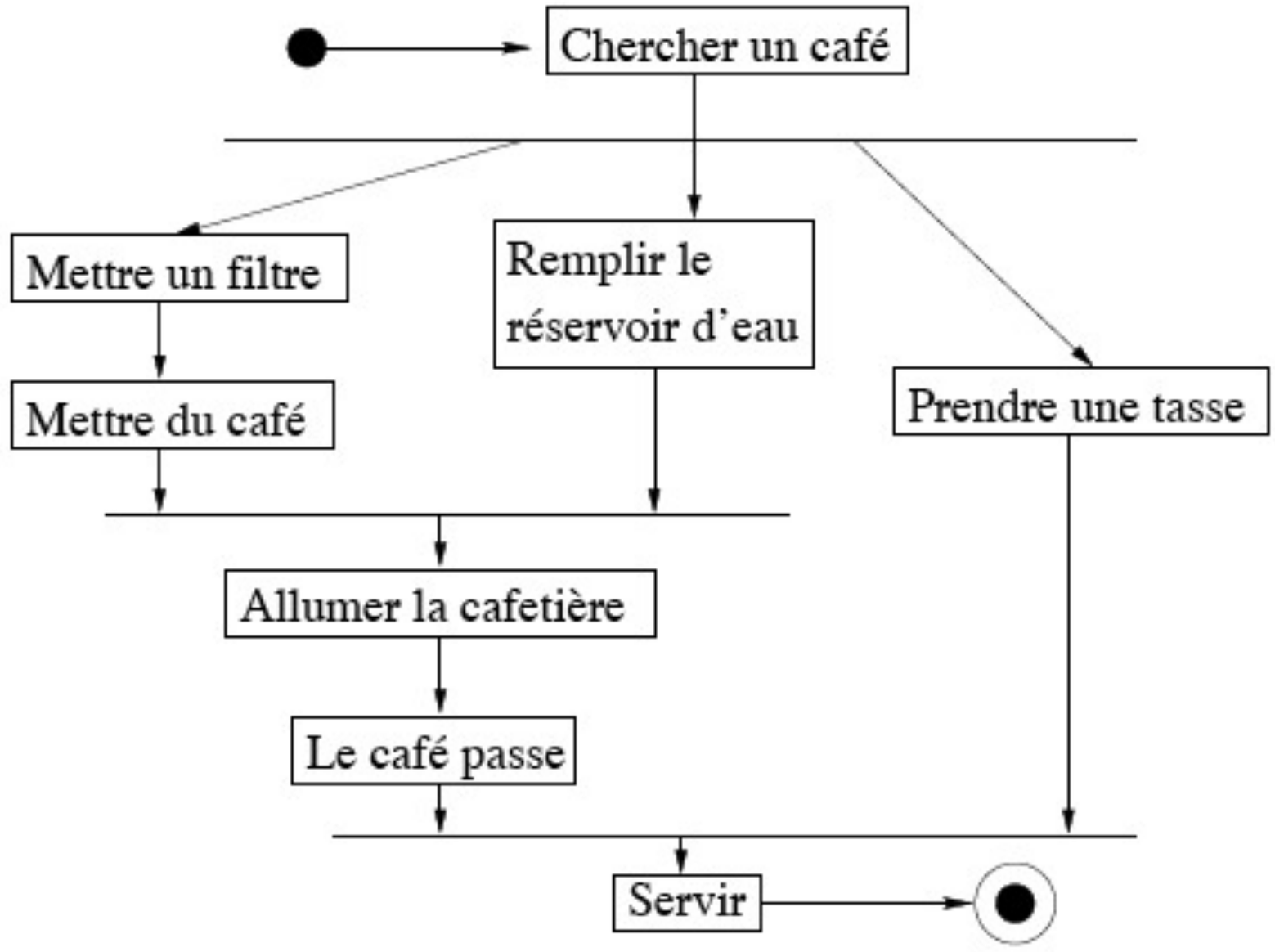
http://wwwdfr.ensta.fr/Cours/docs/IN204/uml.pdf

Diagrammes d'états



Pour expliciter les attentes d'évènements & les différents états d'un objet.

Diagrammes d'activités



Qu'est-ce qu'UML ?
Divers modes d'utilisation selon [Fowler 2003]

➡ **Mode esquisse (*sketch*)**

- Informelle, incomplète
- Souvent manuelle (tableau)

➡ ***Support de communication pour concevoir les parties critiques***

➡ **Mode plan (*blue print*)**

- Diagrammes détaillés

➡***Génération d'un squelette de code à partir des diagrammes***

➡***Nécessité de compléter le code pour obtenir un exécutable***

➡ **Mode langage de programmation**

- Spécification complète, formelle et **exécutable**

➡***Pas vraiment disponible actuellement !***

Qu'est-ce qu'UML ?
Divers modes d'utilisation selon [Fowler 2003]

➡ **Mode esquisse (*sketch*)**

- Informelle, incomplète
- Souvent manuelle (tableau)

➡ ***Support de communication pour concevoir les parties critiques***

➡ **Mode plan (*blue print*)**

- Diagrammes détaillés

➡***Génération d'un squelette de code à partir des diagrammes***

➡***Nécessité de compléter le code pour obtenir un exécutable***

➡ **Mode langage de programmation**

- Spécification complète, formelle et **exécutable**

➡***Pas vraiment disponible actuellement !***

Qu'est-ce qu'UML ?
Divers modes d'utilisation selon [Fowler 2003]

➡ **Modèles descriptifs vs prescriptifs**

- Descriptifs ; Décrire l'existant (domaine, métier)
- Prescriptifs ; Décrire le futur système à réaliser

➡ **Modèles destinés à différents acteurs**

- Pour l'utilisateur ; Décrire le quoi
- Pour les concepteurs/développeurs ; Décrire le comment

http://liris.cnrs.fr/csolnon/coursUML.pdf

Vos objectifs

- Savoir modéliser un problème :
  - En répondant aux besoins utilisateurs
  - En assurant la qualité du logiciel produit
- Connaître la modélisation UML
  - Savoir lire et construire des modèles
  - Faire le lien entre modèle et code

Mes objectifs

- Vous faire le lien entre UML et GL
- Les deux sont **complémentaires** et non pas contradictoires
- La conception et l'analyse s'intègrent dans les méthodes de développement agile

Organisation

- TD en lien avec le projet de GL (mais pas forcément toujours !)
- QCM surprises parfois en TD
- Examen « cas d'usage »
- TD machine : visual paradigm