Bases de la conception orientée objet

À destination des étudiants de
1e année IUT (S2)

Mireille Blay-Fornarino
Université Nice Sophia Antipolis
blay@unice.fr
http://mireilleblayfornarino.i3s.unice.fr/

Site web du module :
https://mbf-iut.i3s.unice.fr/

Objectifs du cours

- **Savoir modéliser un problème pour ensuite pouvoir l'implémenter**
  - En répondant aux besoins des utilisateurs
  - En assurant la qualité du logiciel produit (performance, utilisabilité, sécurité, maintenabilité, ...)
- **Connaître la modélisation UML**
  - savoir lire des modèles ; savoir les construire
  - Faire le lien entre un modèle et le code qui pourrait correspondre.

VOTRE OBJECTIF ?

Atteindre les objectifs du cours?

Avoir la meilleure note possible à cette UE?

Comment?

écouter, essayer, demander,

RESPECTER LES REGLES DONNEES

Notation

- Des notes sur des rendus en TD
- Un rendu final type carnet de bord

Tout retard est sanctionné.

- Un examen final portant sur une étude de cas
- Des contrôles surprises

Ne pas respecter les consignes est sanctionné (feuille A4 autorisée)

Pour augmenter la moyenne !

Introduction

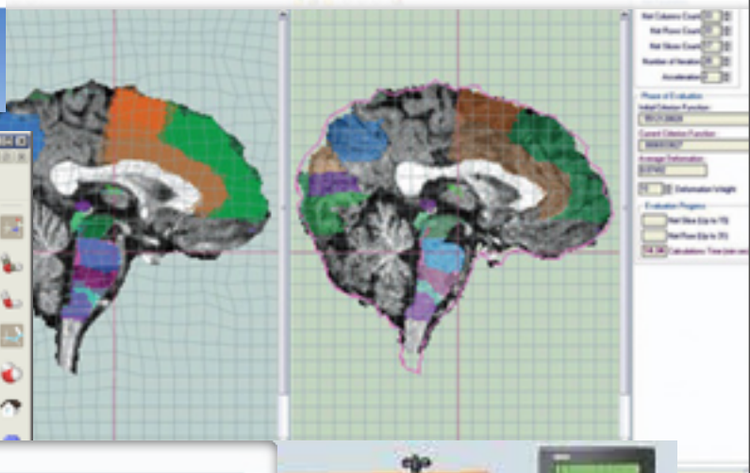
Où se trouve le logiciel ?

Finances

Arts, ..

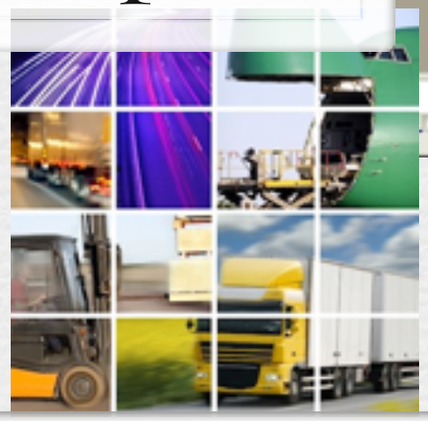
telecommunication

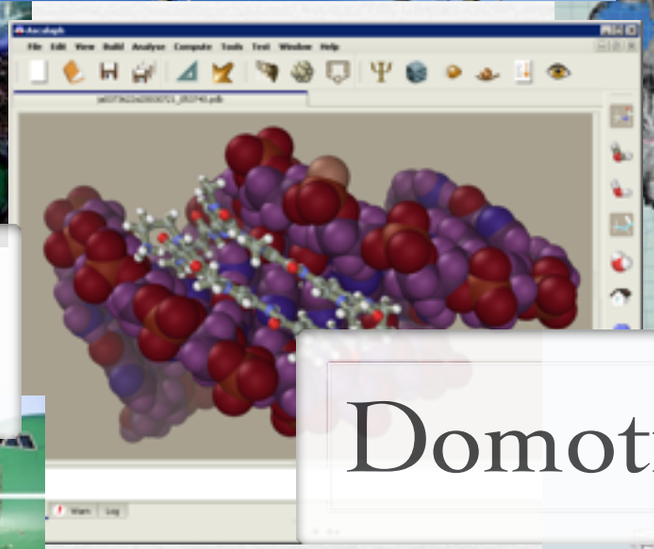
Medicine



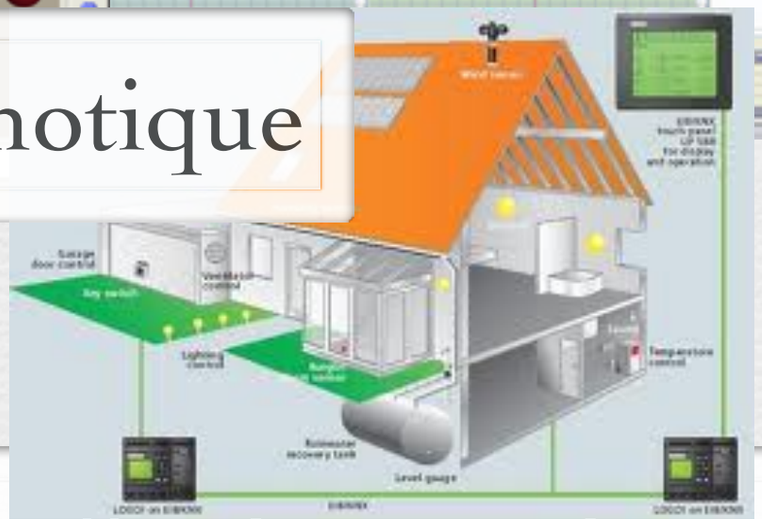
Industrie

Transports





Domotique



Web

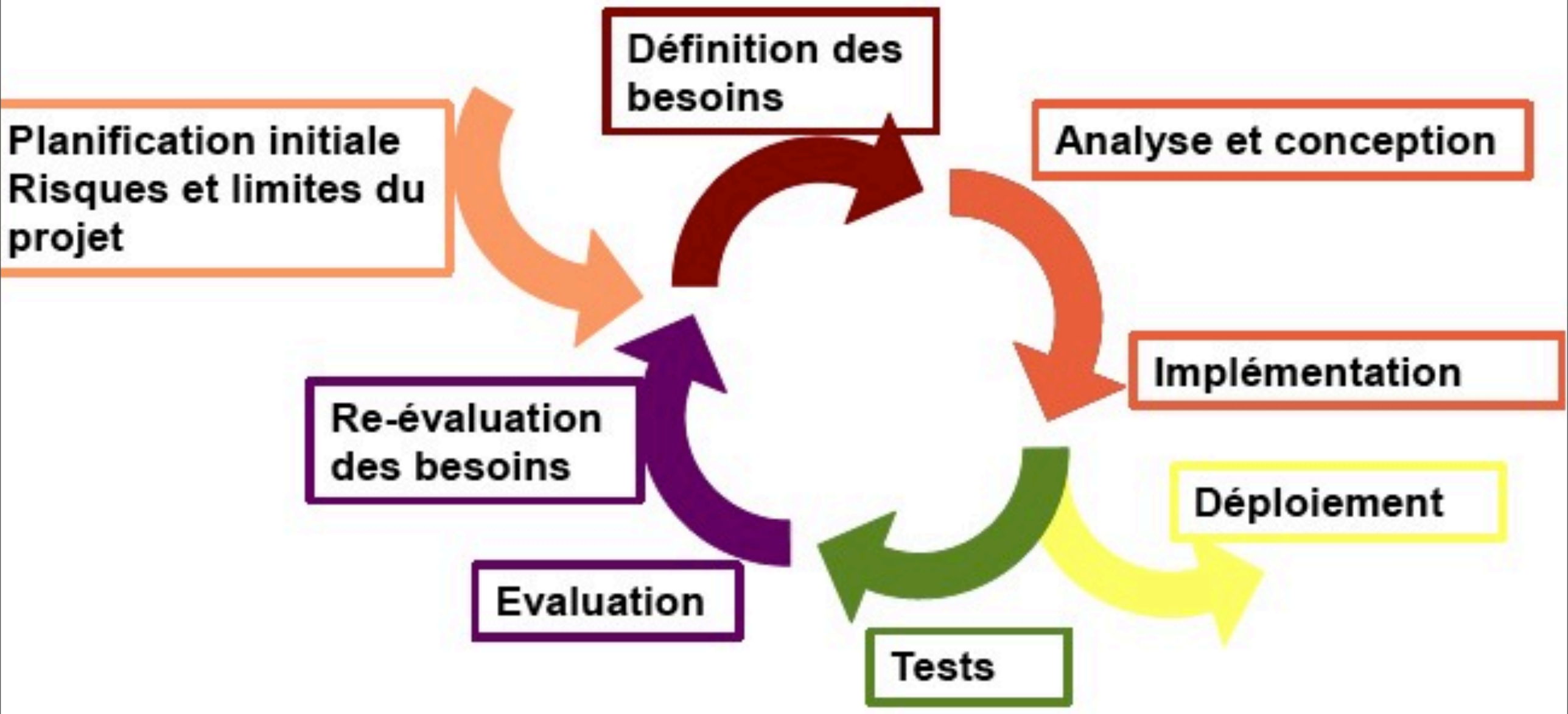
Fouille de données

Nouvelles interactions

En musique, sport, ...

Vous devez réaliser une application logicielle pour visualiser les données météo dans un cockpit, que faîtes-vous?

Activités du **développement** logiciel



La programmation n'est qu'une étape du développement !!

Extrait de Bonnes Pratiques de développement
ENVOL_2010: 27 Septembre
Véronique Baudin
Violaine Louvet

Quelle qualité pour le logiciel ?

Monteriez-vous dans l'avion pour lequel vous avez écrit le système de visualisation des données météo?
Celui dont vous avez écrit le logiciel support aux commandes automatiques?

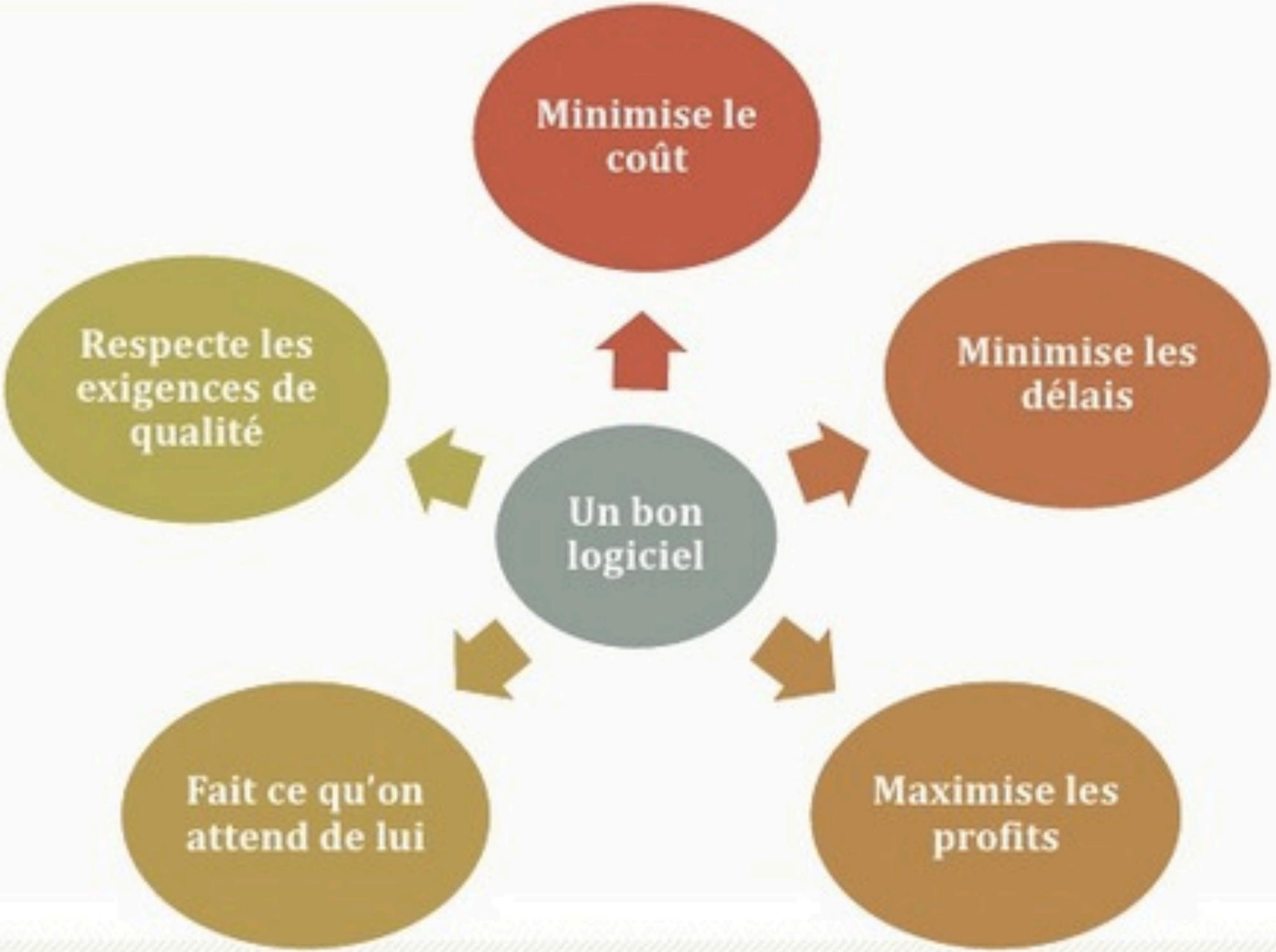
Vous avez râlé sur certains logiciels, ...
Pourquoi?
Etes-vous certains de savoir faire mieux?

Vos pages web... sont-elles de «qualité»?

Un Bon Logiciel du Point de Vue d'un Client

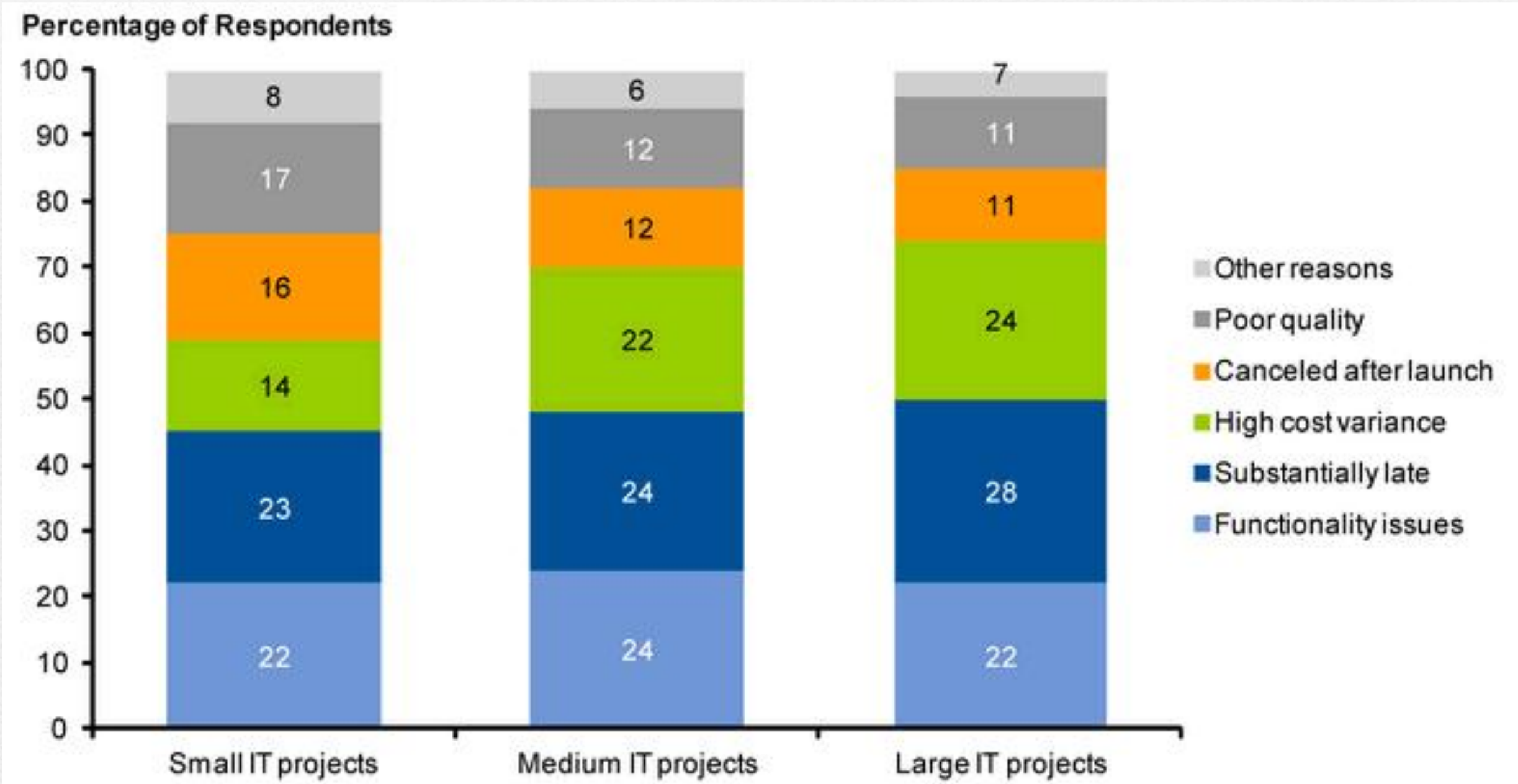
http://fr.slideshare.net/mostefaiamine/cours-gnie-logiciel-introduction

Un Bon Logiciel du Point de Vue d'un Fournisseur



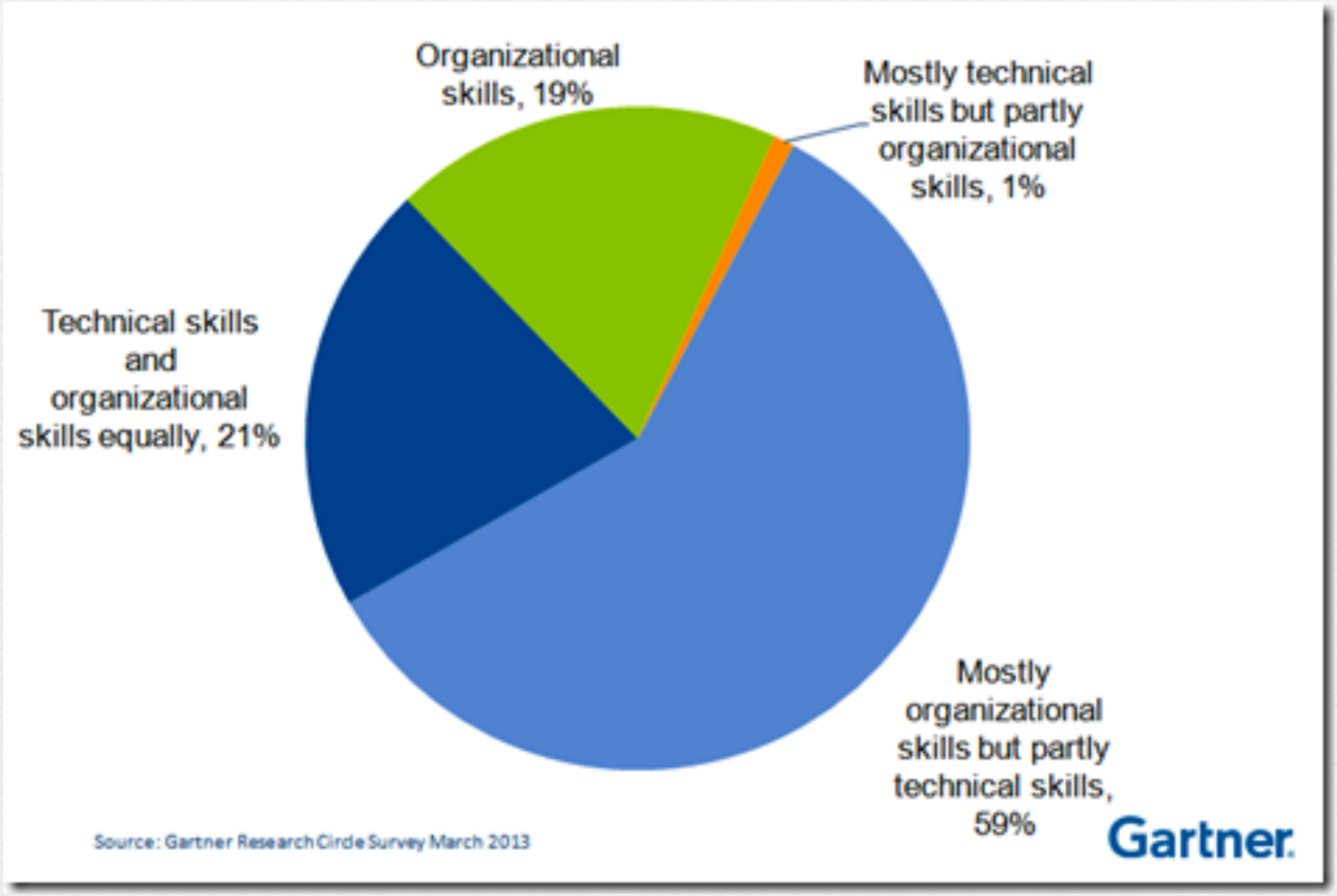
http://fr.slideshare.net/mostefaiamine/cours-gnie-logiciel-introduction

Raisons de l'échec des projets (Gartner 2012)



http://thisiswhatgoodlookslike.files.wordpress.com/2012/06/gartner-fig-2.jpg

Raisons de l'échec des projets (Gartner 2013)



http://blogs.gartner.com/mike-rollings/2013/03/28/why-projects-fail-hint-its-not-technical-skills/

Plan

- Problèmes du développement logiciel
  - Histoire brève jusqu'aux limites de la programmation structurée
  - Du bidouillage au Génie logiciel
- Introduction à UML
  - Un peu d'histoire
  - Survol
- Présentation du Module : démarche générale

I. Quels sont les problèmes du développement logiciel?

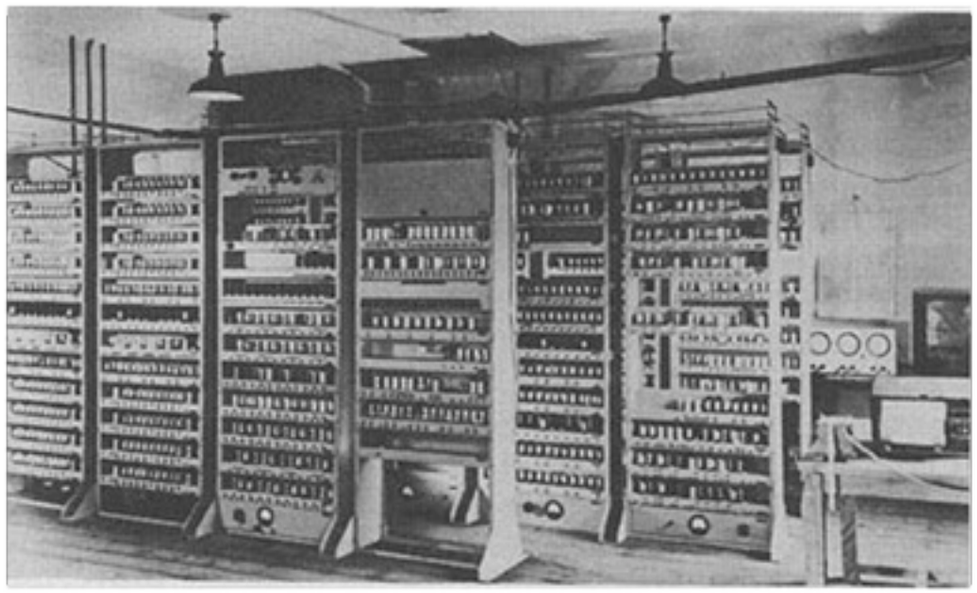
I. Quels sont les problèmes du développement logiciel?

Un peu d'histoire

La gestion progressive de la complexité

Premiers programmes en langage machine

- Les premiers programmes, écrits en langage machine, **dépendaient fortement de l'architecture** des ordinateurs utilisés.



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

La gestion progressive de la complexité

Programmation en langages évolués

- Lorsque le nombre d'architectures différentes a augmenté, un premier effort a été produit pour **séparer les concepts manipulés dans les langages de leur représentation dans la machine** et a abouti à la création de langages comme FORTRAN.

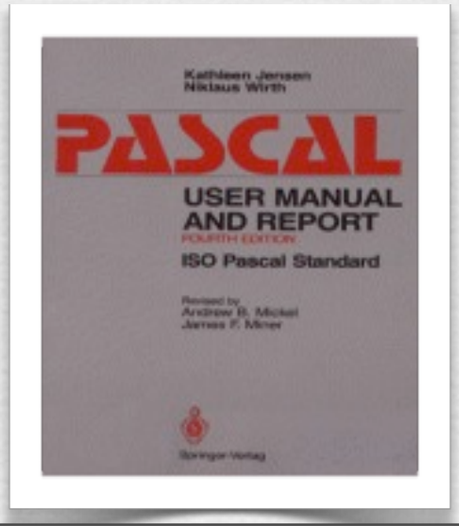

```
      PROGRAM DEGRAD
!
! Imprime une table de conversion degrés -> radians
! ================================================
!
! Déclaration des variables
      INTEGER DEG
      REAL RAD, COEFF
!
! En-tête de programme
      WRITE ( *, 10)
   10 FORMAT       (' ',20('*') /                &
     &              ' * 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
```

La gestion progressive de la complexité

Méthode d'analyse par décomposition

- La complexité croissante des programmes a de nouveau suscité un effort pour mieux **structurer les programmes** (abandon du goto et de la programmation spaghetti)
- Les méthodes d'analyse consistait alors à «diviser pour mieux régner», i.e. à **découper les tâches en modules indépendants**.
- Niklaus Wirth va plus loin : les programmes sont la «somme» d'une structure de données et d'un ensemble distinct d'algorithmes chargés de les manipuler.

➡ La programmation structurée étaient alors synonyme de **programmation dirigée par les traitements**.



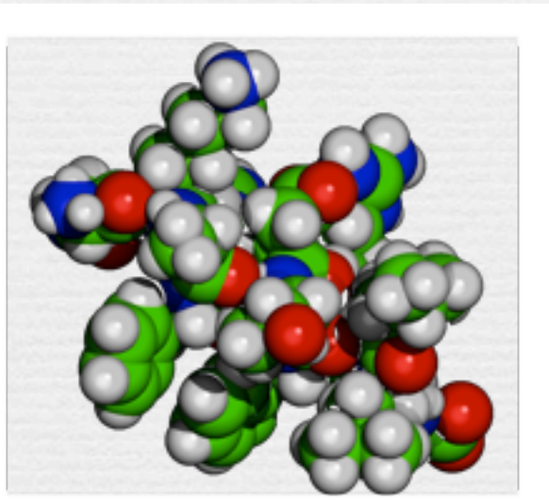

```
function ConstruitPhrase(phrase:string):string;
var s : string;
begin
  readln(s);
  ConstruitPhrase:=phrase+', '+s;
end;

var
  Phrase : string;
begin
  Phrase:='';  {initialiser à chaîne vide}
  {premier mot}
  writeln('Entrez un mot :');
  Phrase:=ConstruitPhrase(Phrase);
```

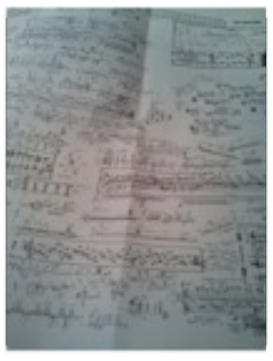
La gestion progressive de la complexité

L'explosion des besoins

- Le coût du matériel informatique a fortement décru et ce matériel est devenu un bien de consommation courant (tant dans des entreprises et des universités que chez les particuliers).
- La clientèle s'est diversifiée, les **besoins ont explosé**.



Quelles propriétés ces logiciels doivent respecter d'après vous?
Que feriez-vous?



Et après

Encapsuler, assembler, réutiliser

- Pour répondre à ces besoins croissants, la programmation a connu une évolution et une montée en abstraction encore plus importante :

  Objets, **Composants**, **Services**, **Composants as Services**, Génération des codes à partir de modèles, **Frameworks**, **Usines logicielles**, **DSL** ...

- Intensification des tests automatiques, des approches «formelles», ...

Mais aussi des changements de

«méthodes» de développement...

I. Quels sont les problèmes du développement logiciel?

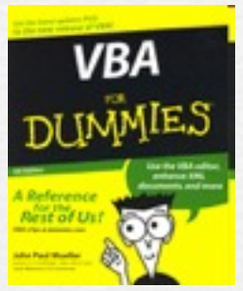
Du bidouillage au génie logiel

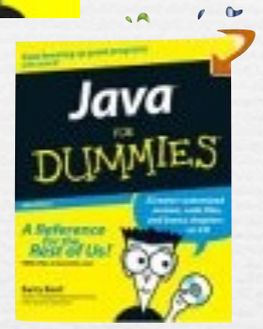
Problématique du génie logiciel

- Programming-in-the-Small
  - Problème de la **qualité** interne d'un composant
- Programming-in-the-Large
  - Faire face à la construction de systèmes de plus en plus gros : non spécifique au logiciel, mais aggravé par sa “mollesse”.
  - Problème de gestion de la complexité, de communication, etc.
  - Maîtrise du *processus de développement*: délais, coûts, qualité.
- Programming-in-the-Duration
  - Problème de la maintenance corrective et évolutive.
  - Notion de ligne de produits.

Programming-in-the-Small

Problème de la validité du logiciel






```
Acquérir une valeur positive n
Tant que n > 1 faire
si n est pair
alors n := n / 2
sinon n := 3n+1
Sonner alarme;
```

Prouver que le prog. termine pour tout n?
-> Indécidabilité de certaines propriétés

Recours au test
– ici, si machine 32 bits, 2^31 = 10^10 cas de tests
– **5 lignes de code => *10 milliards de tests* !**

Ou à la preuve => de savoir exprimer ce que l'on veut

Programming-in-the-Large

Gérer la complexité

a)

c)

b)

d)

Si l'automobile avait suivi le même développement que l'ordinateur, une Rolls-Royce coûterait aujourd'hui 100$, pourrait rouler un million de kilomètres avec un litre d'essence, et exploserait une fois par an en tuant tout le monde à bord. *Robert Cringely.*

Combien de lignes de codes

World of War Craft ?

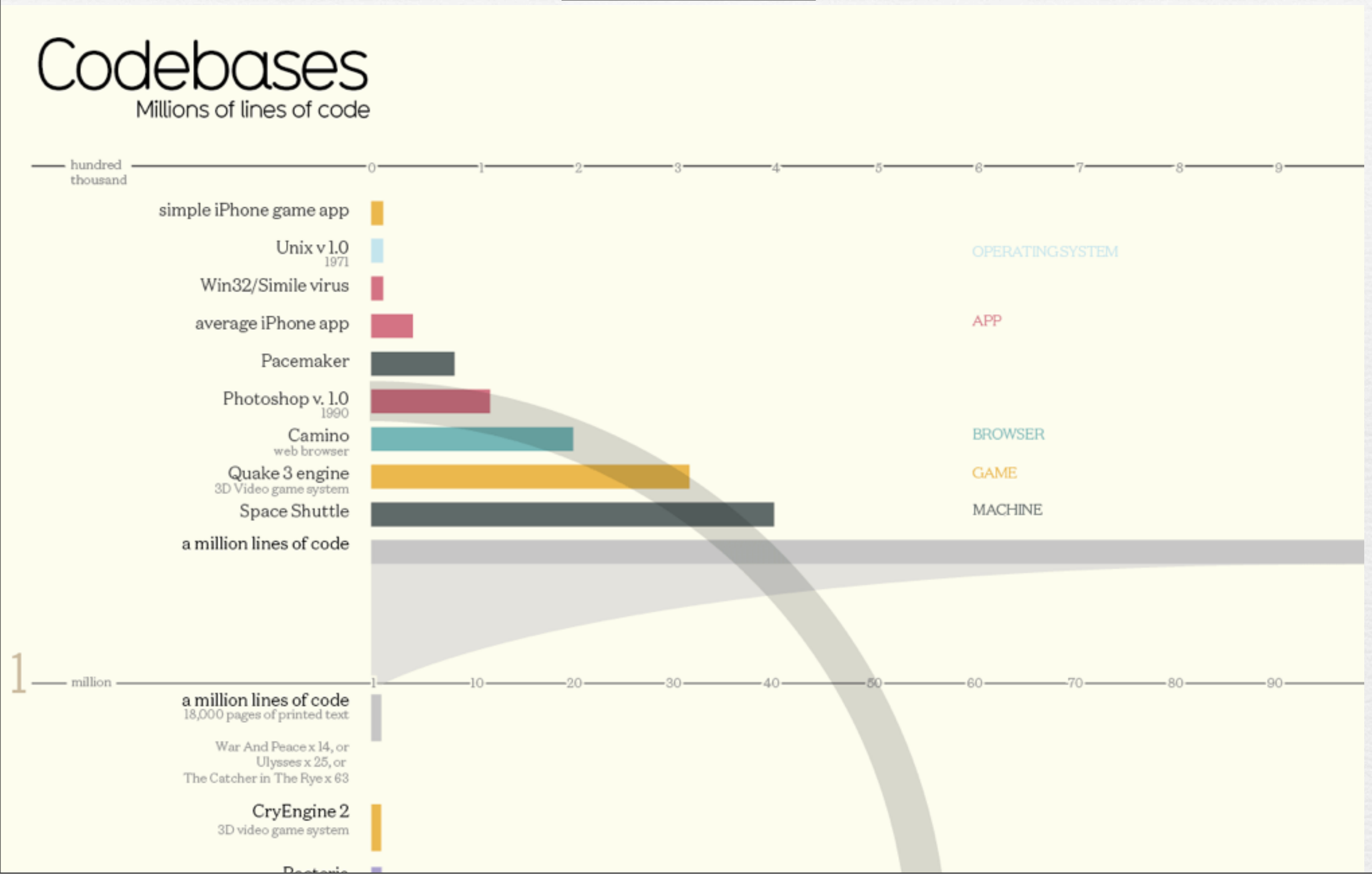
Une simple application de jeu sur Iphone représente moins d'une centaine de milliers de lignes de code. Quand le navigateur web Google Chrome ou le jeu en ligne *World of WarCraft* se chiffrent à près de cinq millions.

Facebook ?

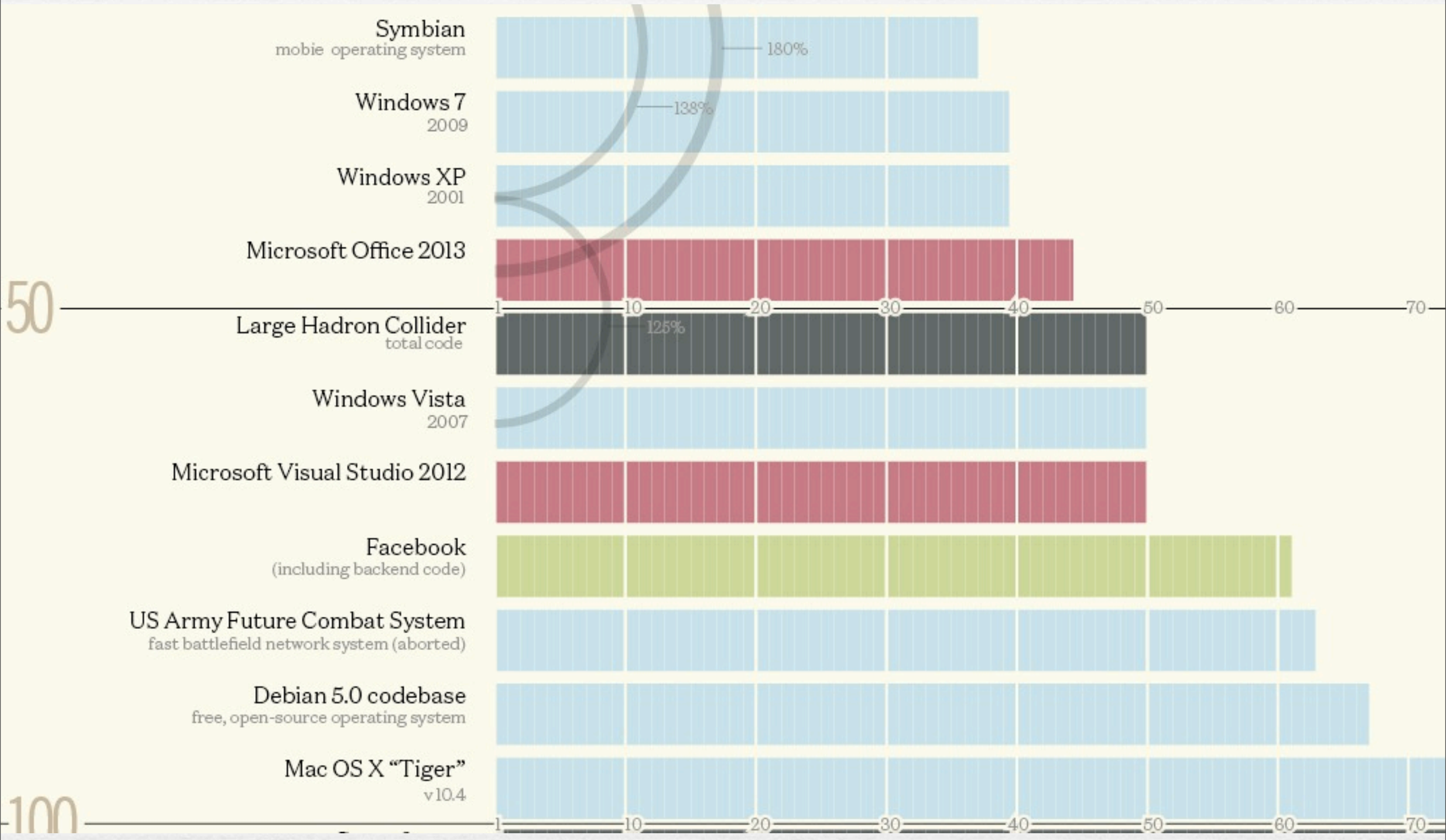
Et que dire du réseau social le plus utilisé dans le monde, Facebook, qui a eu recours à plus de 60 millions de lignes de code, derrière le dernier logiciel d'Apple, le Mac OS X "Tiger" avec plus de 80 millions.

http://www.bfmtv.com/high-tech/combien-lignes-codes-facebook-windows-wow-633862.html

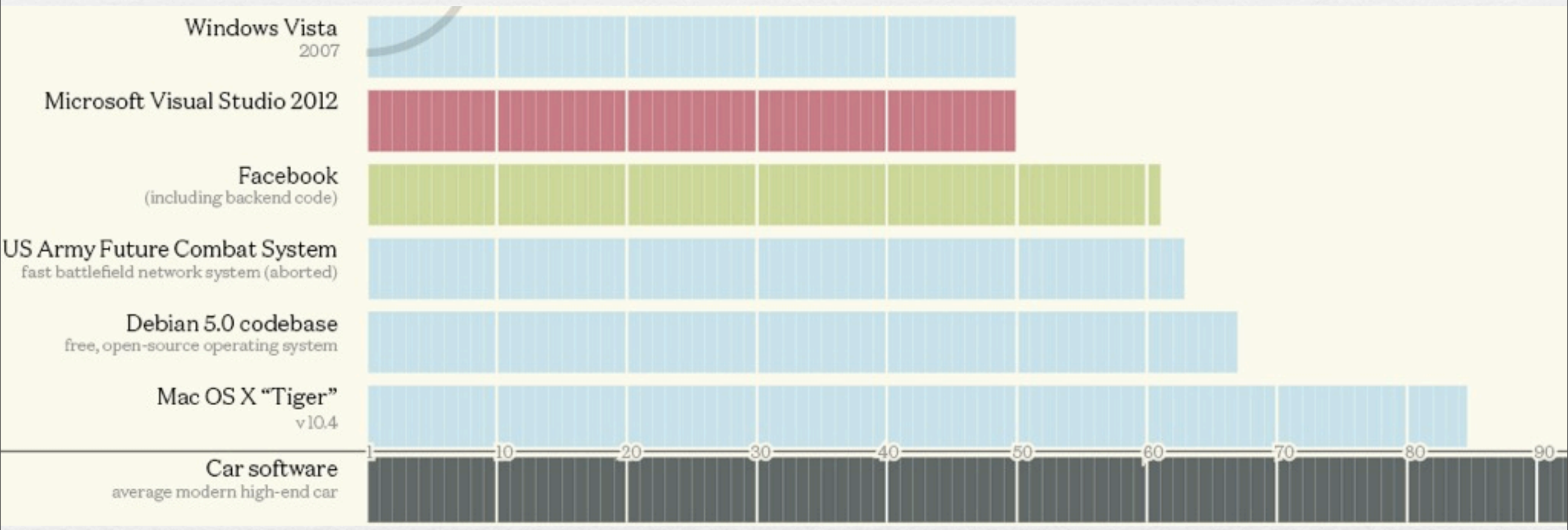
http://www.informationisbeautiful.net/visualizations/million-lines-of-code/



http://www.informationisbeautiful.net/visualizations/million-lines-of-code/



http://www.informationisbeautiful.net/visualizations/million-lines-of-code/



Programming-in-the-Large

Echecs logiciels

- ATT (1990):
  interruption du service téléphonie
  pendant 5 heures à l'Est des Etats-Unis.
- Aéroport de Denver, 1993-1995:
  logiciel de suivi des bagages,
  coût 3 milliards de $, retard de 18 mois.
- National Cancer Institute, Panama City,2000 : **Logiciel non adapté aux médecins => 8 morts, 20 personnes "surexposées".**
- etc., etc.

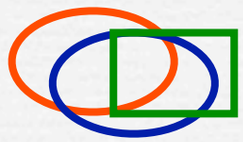
Premier vol d'Ariane 5 (1996)

Out of range

Gestion de la complexité due à la taille : diviser pour résoudre

- Aspects techniques
  - en s'appuyant techniquement sur la modularité
    - « Encapsulation et masquage d'information, faible couplage »
    - Importance de la notion *d'Objets*
- Aspects organisationnels
  - S'organiser au delà de la réalisation : méthodes
    - « Analyse, conception »
    - « Validation et vérification »
  - Ingénierie de la conduite de projet = compromis entre
    - respect des délais
    - respect des coûts
    - réponse aux besoins/assurance qualité

Programming-in-the-Duration



Gestion de l'évolution d'un système

- D'après Swanson & Beath (Maintaining Information Systems in Organizations, 1989)
  - Durée de vie moyenne d'un système : 6.6 ans
  - 26% des systèmes sont âgés de plus de 10 ans
- Problème de la maintenance corrective et évolutive
  - Adaptation aux changements des besoins
  - Adaptation aux changements de l'environnement
    - Support nouvelles plateformes, bug « an 2000 »

Exemple Windows Vista …

Vista prêt à en découdre avec les failles de sécurité!

C'est inévitable*. La découverte de failles devrait s'accélérer avec le lancement en grandeur réelle de Windows Vista. Microsoft s'y prépare…*

…« Les failles sont inhérentes au développement logiciel. Elles reposent sur des erreurs de programmation. Or, détecter absolument toutes les erreurs est ***impossible*** *même avec des moyens financiers comme ceux dont dispose Microsoft »…*

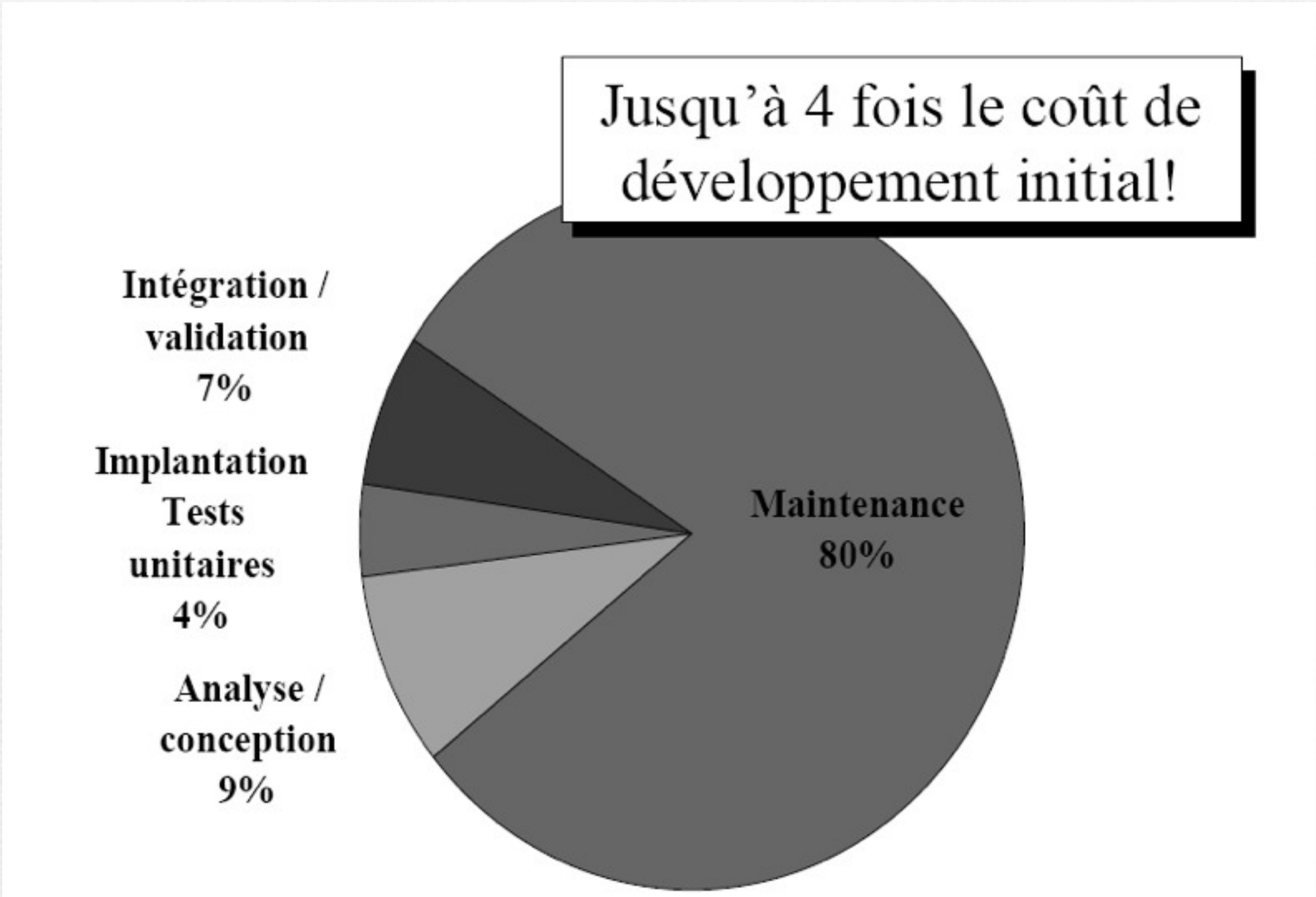
8 000 dollars pour chaque trou de sécurité!

*… **Idefense Labs**, une filiale de **Verisign**, va encore plus loin. Elle promet 8 000 dollars (assortis d'un bonus oscillant entre 2 000 et 4 000 dollars selon la qualité des explications fournies) à qui découvrira une faille critique dans Vista ou Internet Explorer 7…*

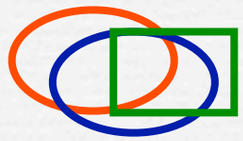
… « La démarche est pertinente dans la mesure où elle permet de mobiliser les chercheurs du monde entier à moindre frais »…

Programming-in-the-Duration

Coût de la Maintenance



Programming-in-the-Duration



Solution : approche par modélisation

- Aspects techniques
  - Assurer une meilleure continuité entre
    - Domaine du problème
    - Domaine des solutions
  - Définir les fonctionnalités
    - Prévoir les scénarios de tests
    - Maintenance traitée comme le développement initial
- Aspects organisationnels
  - Traçabilité
  - Gestion contrôlée des changements

Utilisation de Méthodes «Agiles»

Problématique du «génie (du) logiciel»

WHAT IS SOFTWARE ENGINEERING?

The IEEE Computer Society defines software engineering as

"(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in (1)."

http://www.swebok.org/

III. Introduction à UML

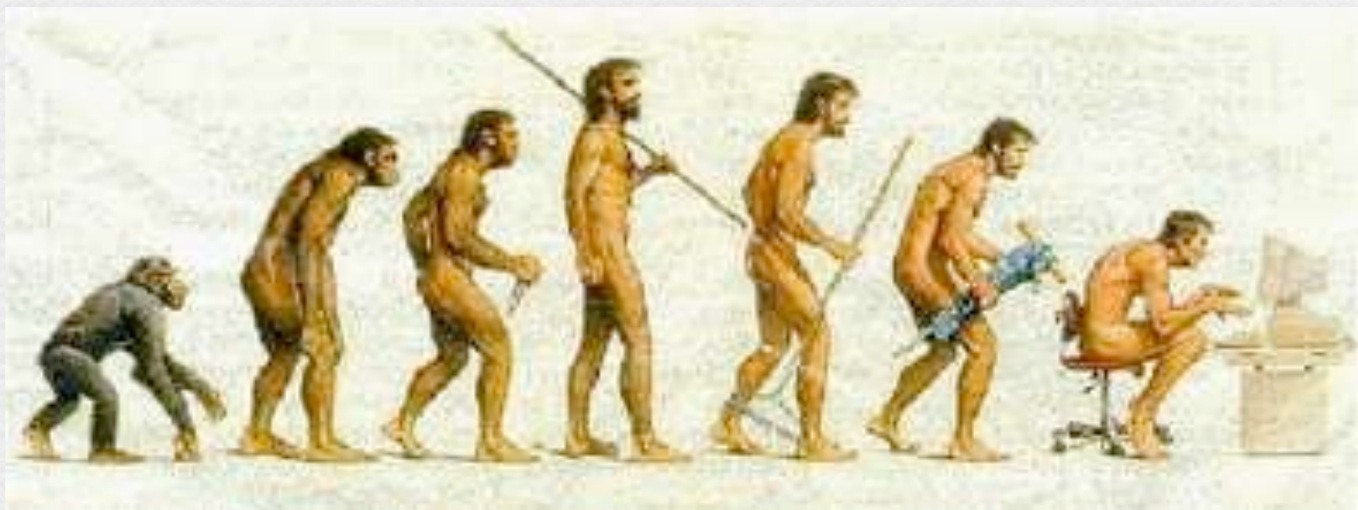
III. Introduction à UML

UML, histoire, généralité

Un peu d'histoire : La guerre des méthodes

Booch, OMT, Coad/Yourdon, Fusion,SADT, OOSE, Schlaer/Mellor, HOOD…

On a au moins besoin d'un «langage» de modélisation standard !



Et un langage unique, un !

Un cocktail de notations éprouvées.
(...mais pas toutes, p. ex. RdP, SADT/IDEF0, DFD, etc.)

- Auteurs : Grady Booch, Ivar Jacobson, James Rumbaugh.
- Standardisation **OMG** (Object Management Group) en 1997
- Promoteurs :
  - Rational Software, Oracle
  - HP, Microsoft, IBM

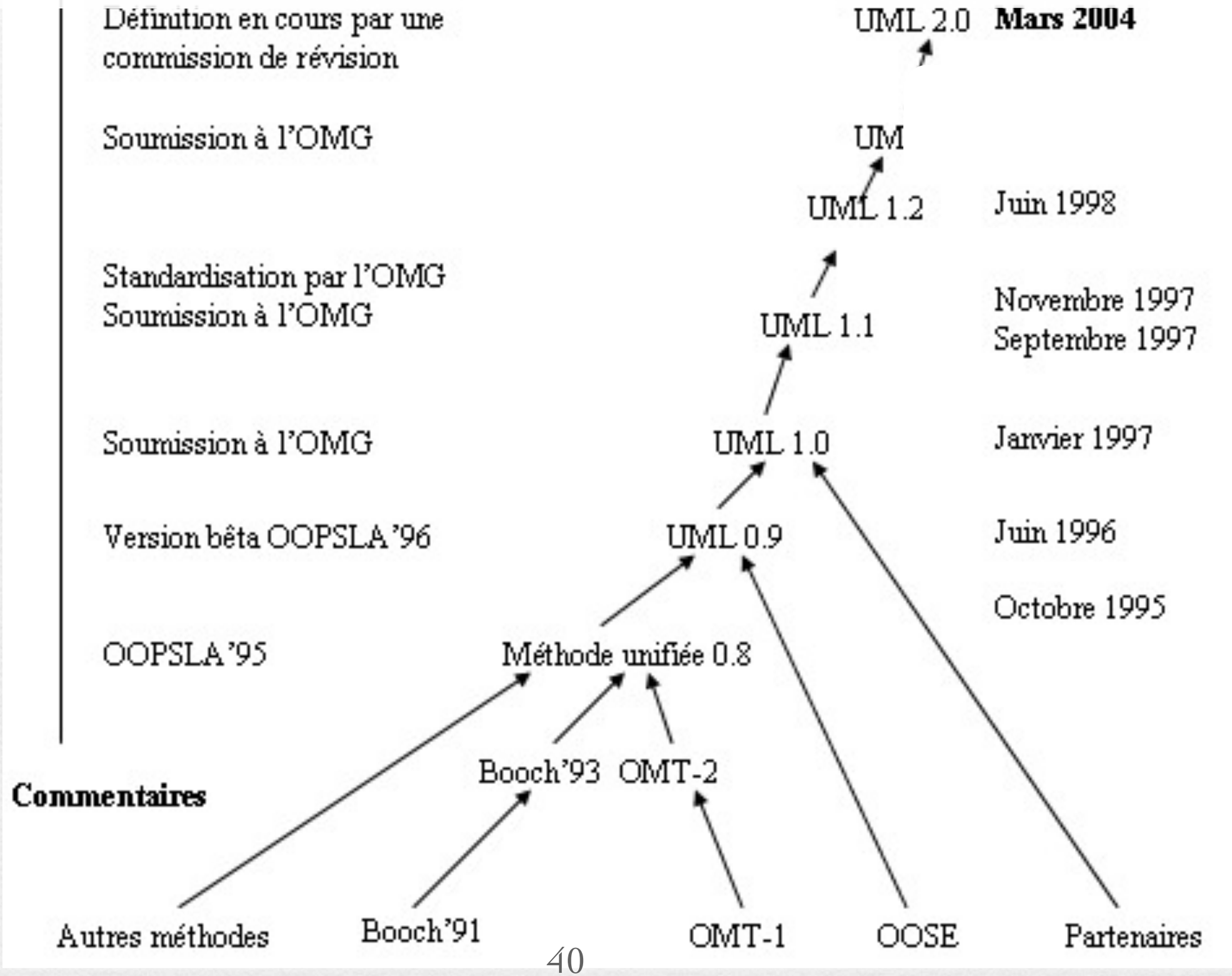
"*Lingua franca*"

http://www.omg.org/spec/UML/2.3/Superstructure/PDF/ 758 pages

UML 2.4 ISO 2012

UML 2.4 août 2011

Commentaires du public

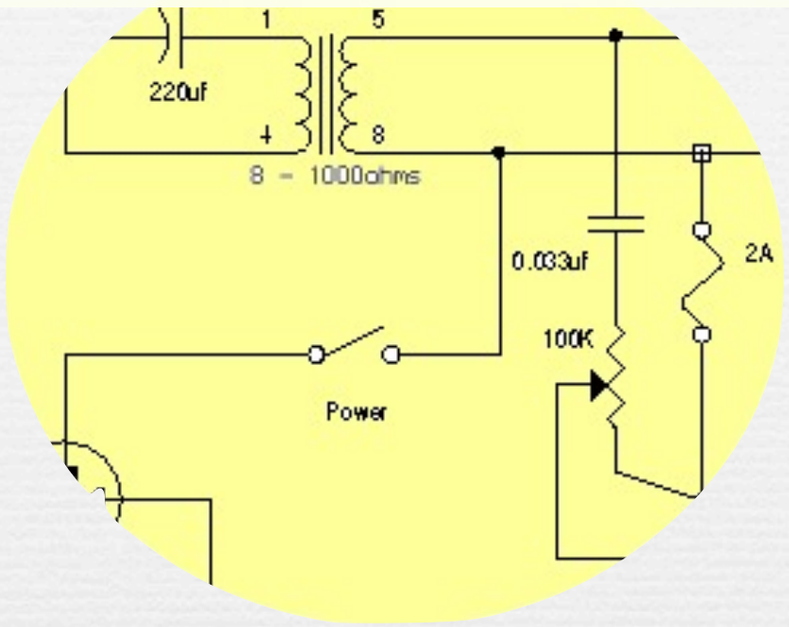


UML dans l'industrie

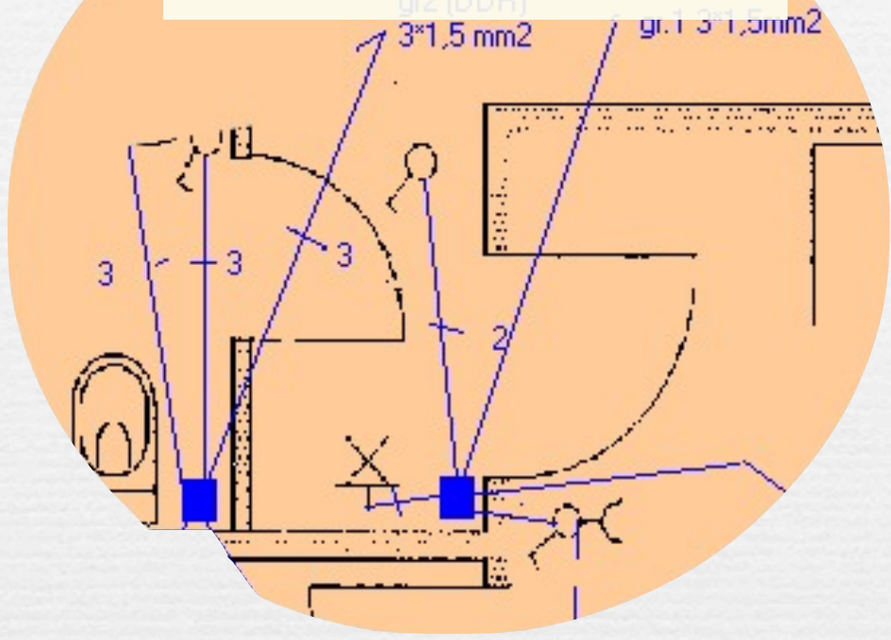
- Sur 500 développeurs : 97% connaissent UML, 56% l'utilisent dans leurs projets de développement, (en 2005)

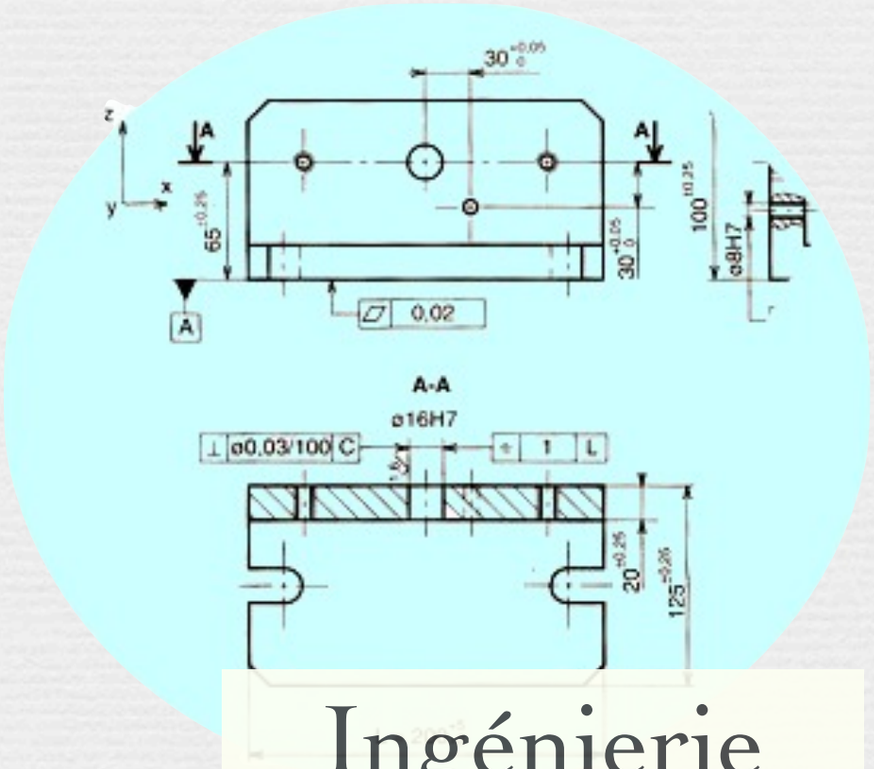
- http://www.prweb.com/releases/2005/04/prweb231386.htm

Ingénierie Électrique

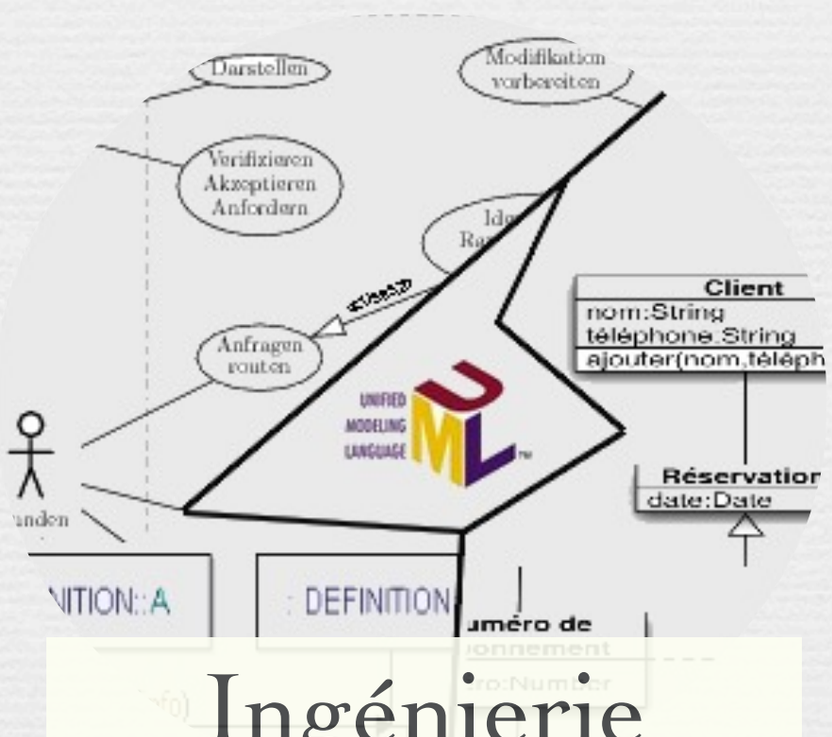


Ingénierie du Bâtiment





Ingénierie Mécanique



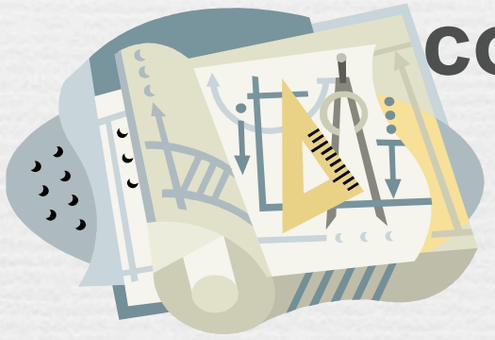
Ingénierie Logicielle

Qu'est-ce qu'UML ?

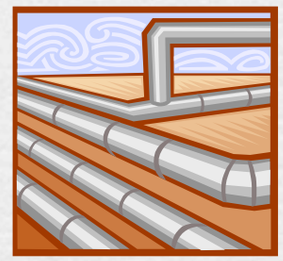
un support à la modélisation

➡ **Modèle** : simplification de la réalité dont les buts sont

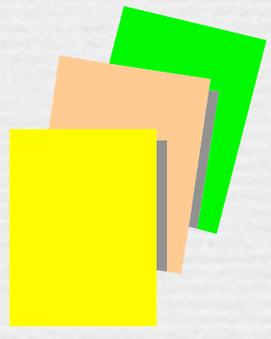
Visualiser
le système



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

- **UML n'est pas une méthodologie de développement, contrairement à RUP (*Rational Unified Process*).**

Les 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
  - Il cadre l'analyse.
  - Il facilite la compréhension de représentations abstraites complexes.
  - Son caractère polyvalent et sa souplesse en font un langage universel.

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

Les points faibles d'UML

- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.

- Le processus de développement (non imposé par UML) est une autre clé de la réussite d'un projet.

III. Introduction à UML

survol

Vue fonctionnelle

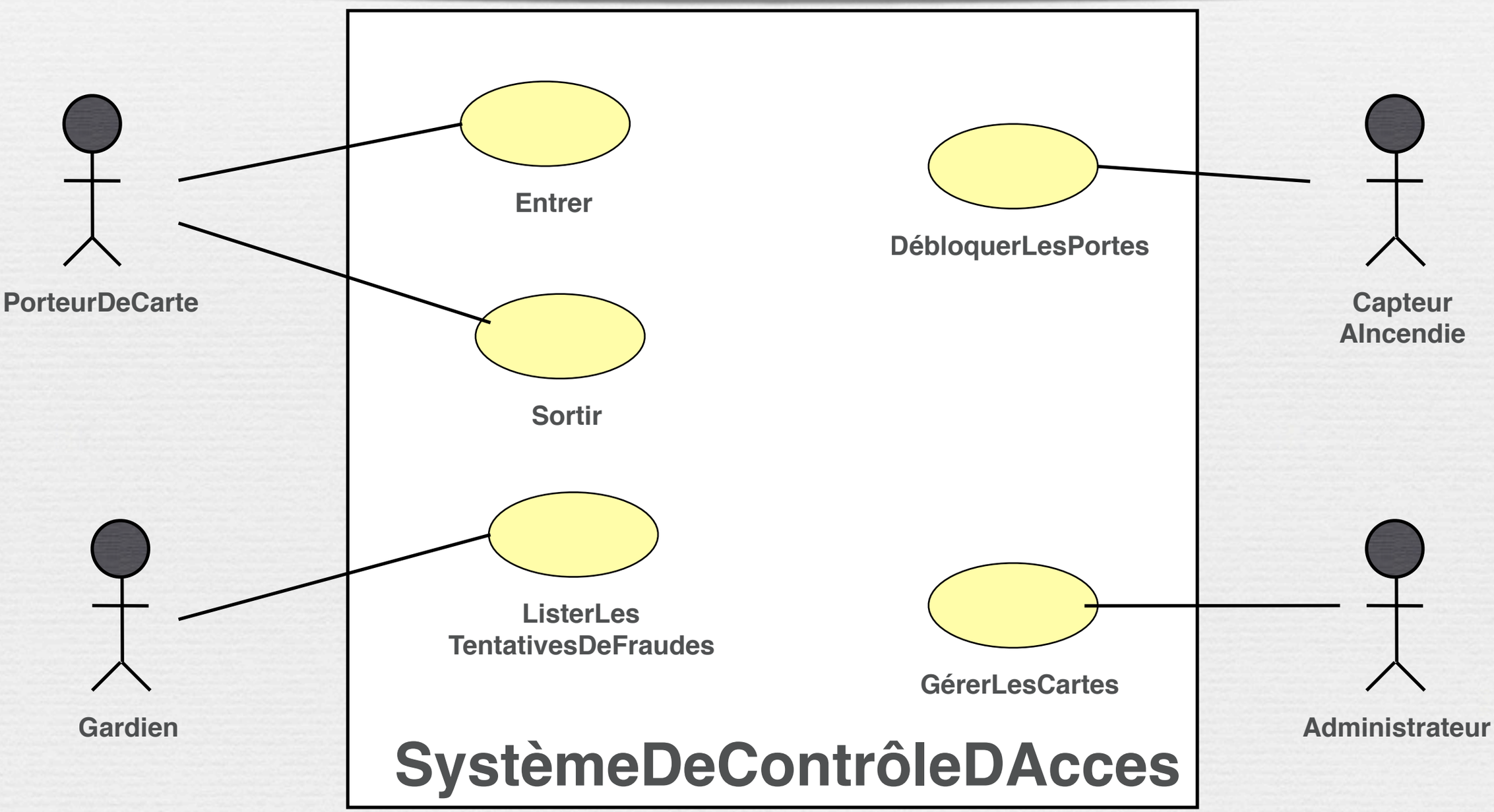
la vue fonctionnelle cherche à appréhender
les interactions entre les
acteurs/utilisateurs
et le système,
sous forme d'objectifs à atteindre (**cas d'utilisation**) et
sous forme chronologique de scénarios d'interaction
typiques (**diagrammes de séquences**)

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

Qu'est-ce qu'UML ?

Diagrammes des cas d'utilisation

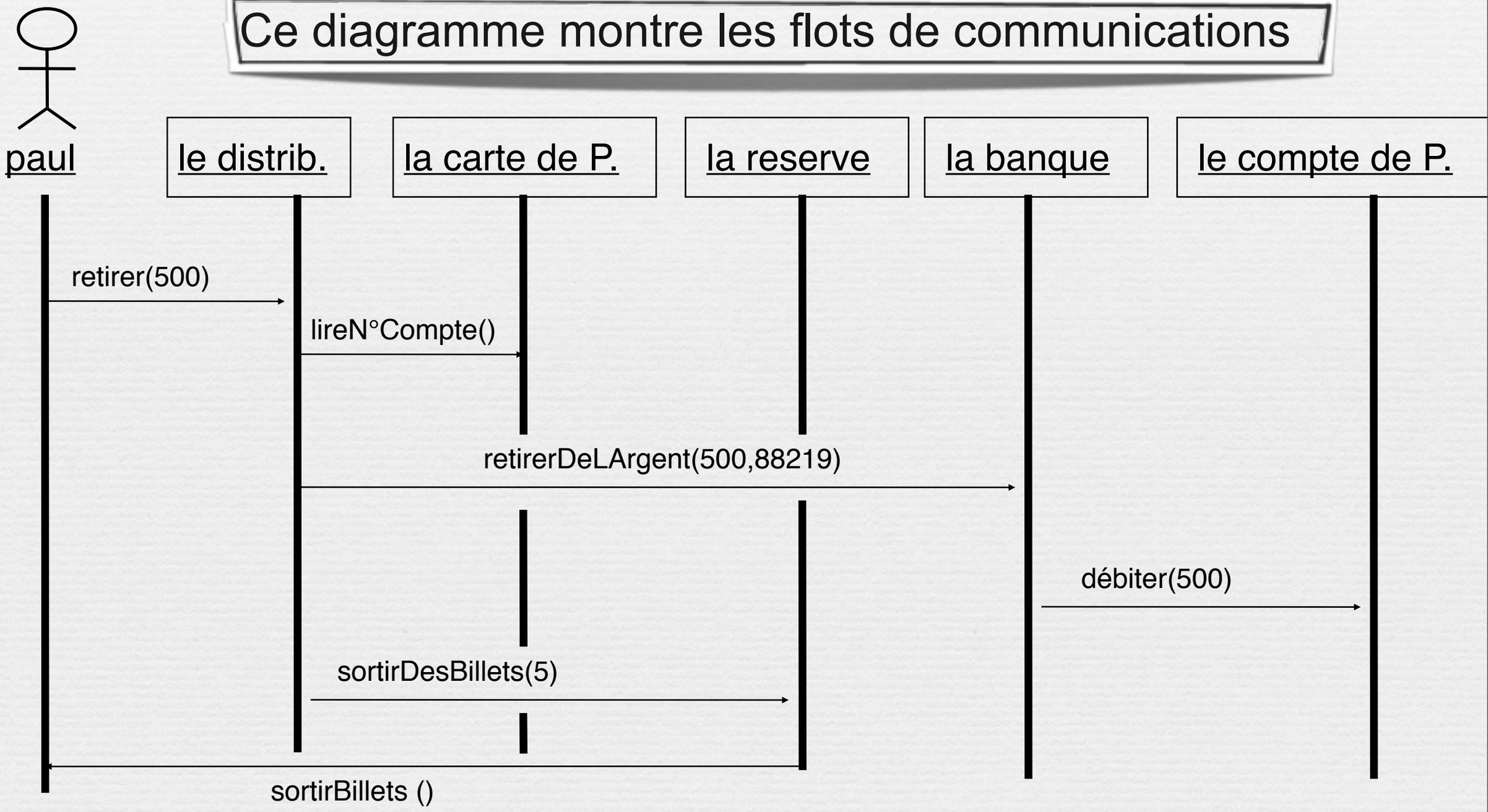
Ce diagramme montre ce que que fait le système et qui l'utilise



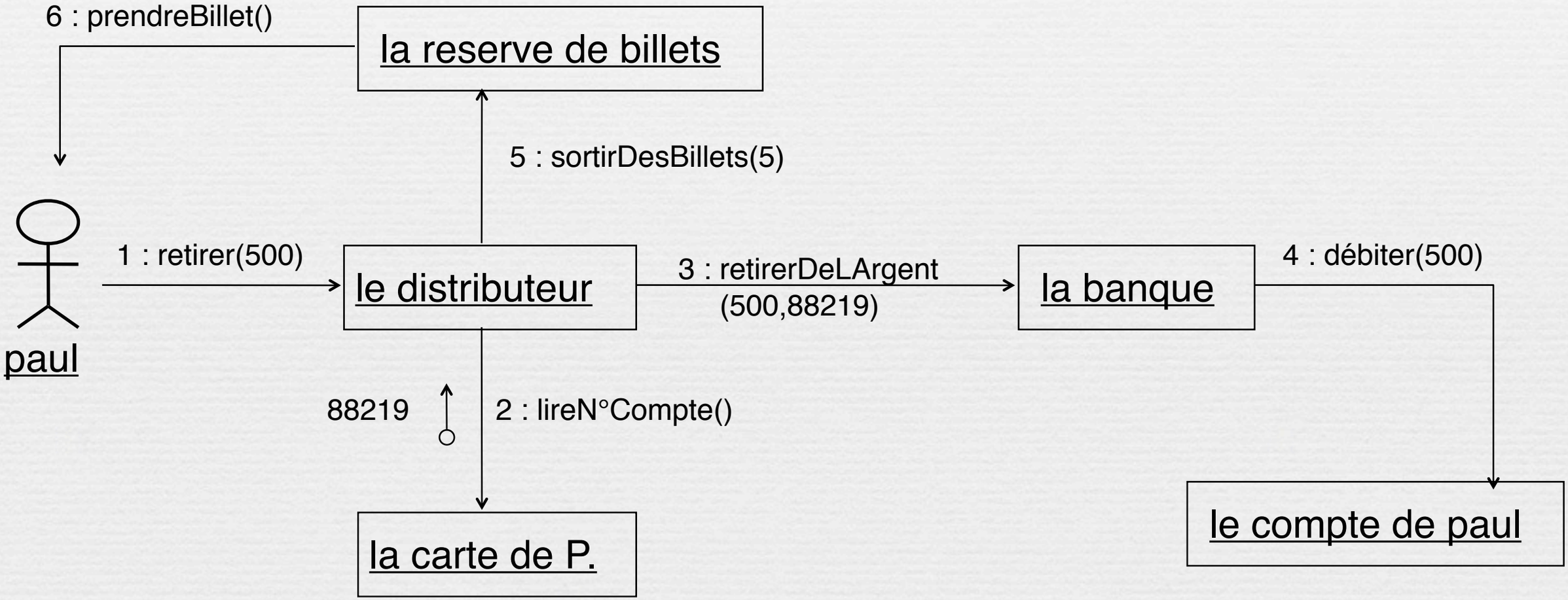
Qu'est-ce qu'UML ?

Diagrammes de séquence

Ce diagramme montre les flots de communications



Qu'est-ce qu'UML ?
Diagrammes de collaboration



Non étudié cette année

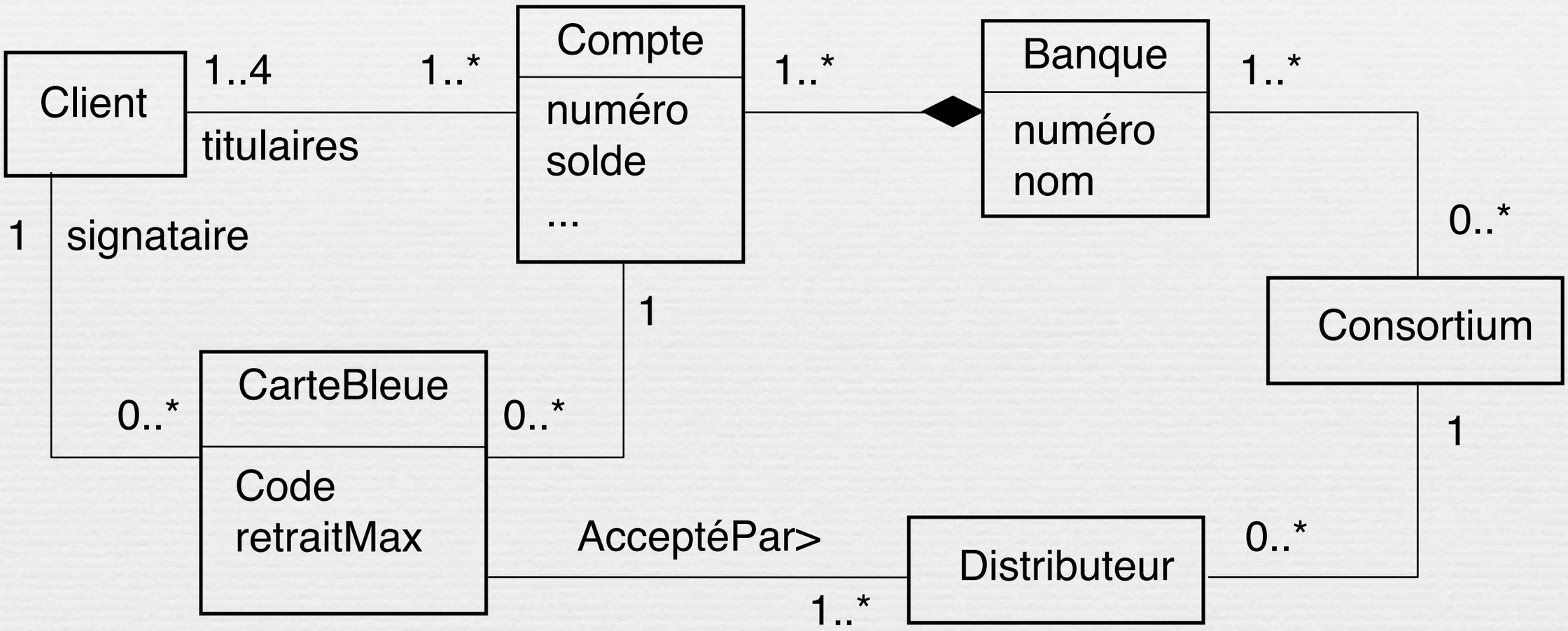
Vue Structurelle

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.

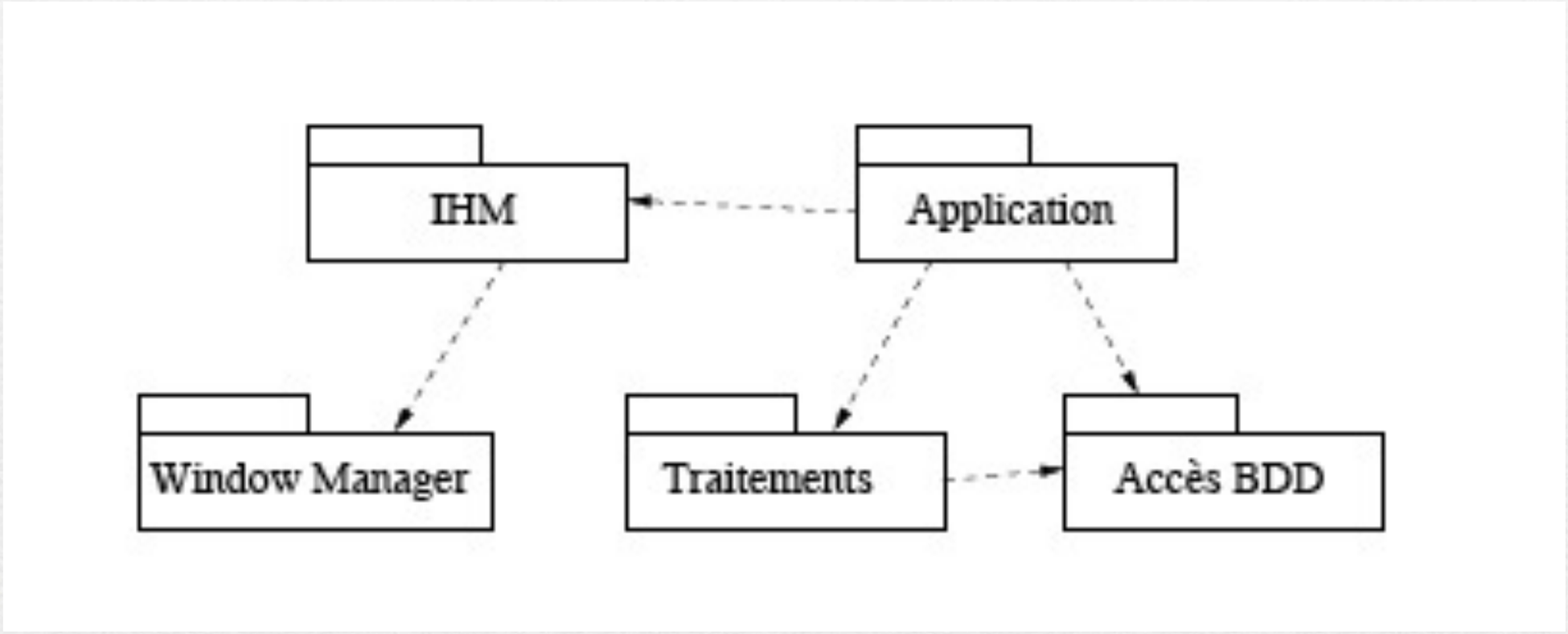
http://wwwdfr.ensta.fr/Cours/docs/IN204/uml.pdf

Qu'est-ce qu'UML ?
Diagrammes de classes



Ce diagramme montre les classes et les relations entre elles

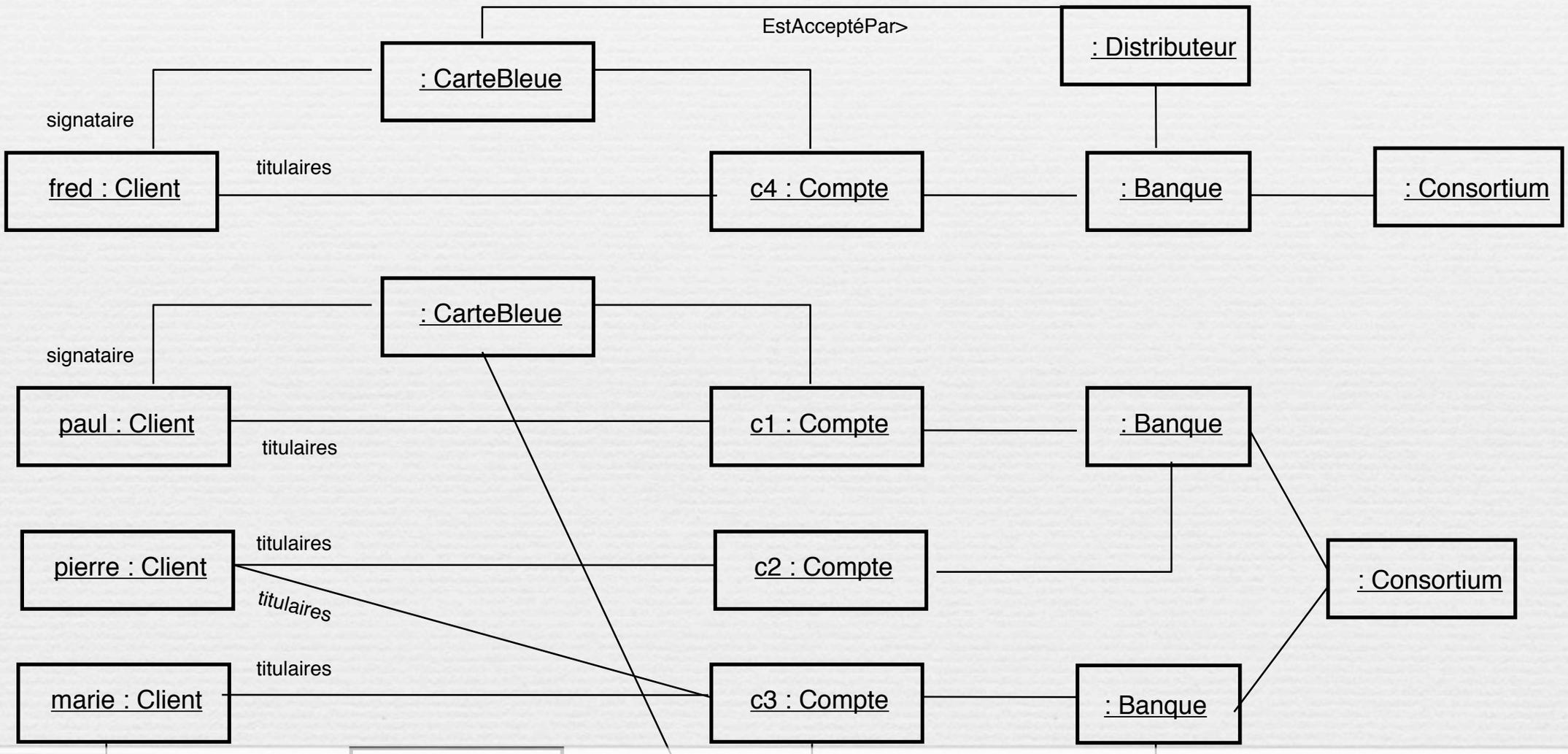
Qu'est-ce qu'UML ?
Diagrammes de packages



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

Qu'est-ce qu'UML ?
Diagrammes d'objets



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

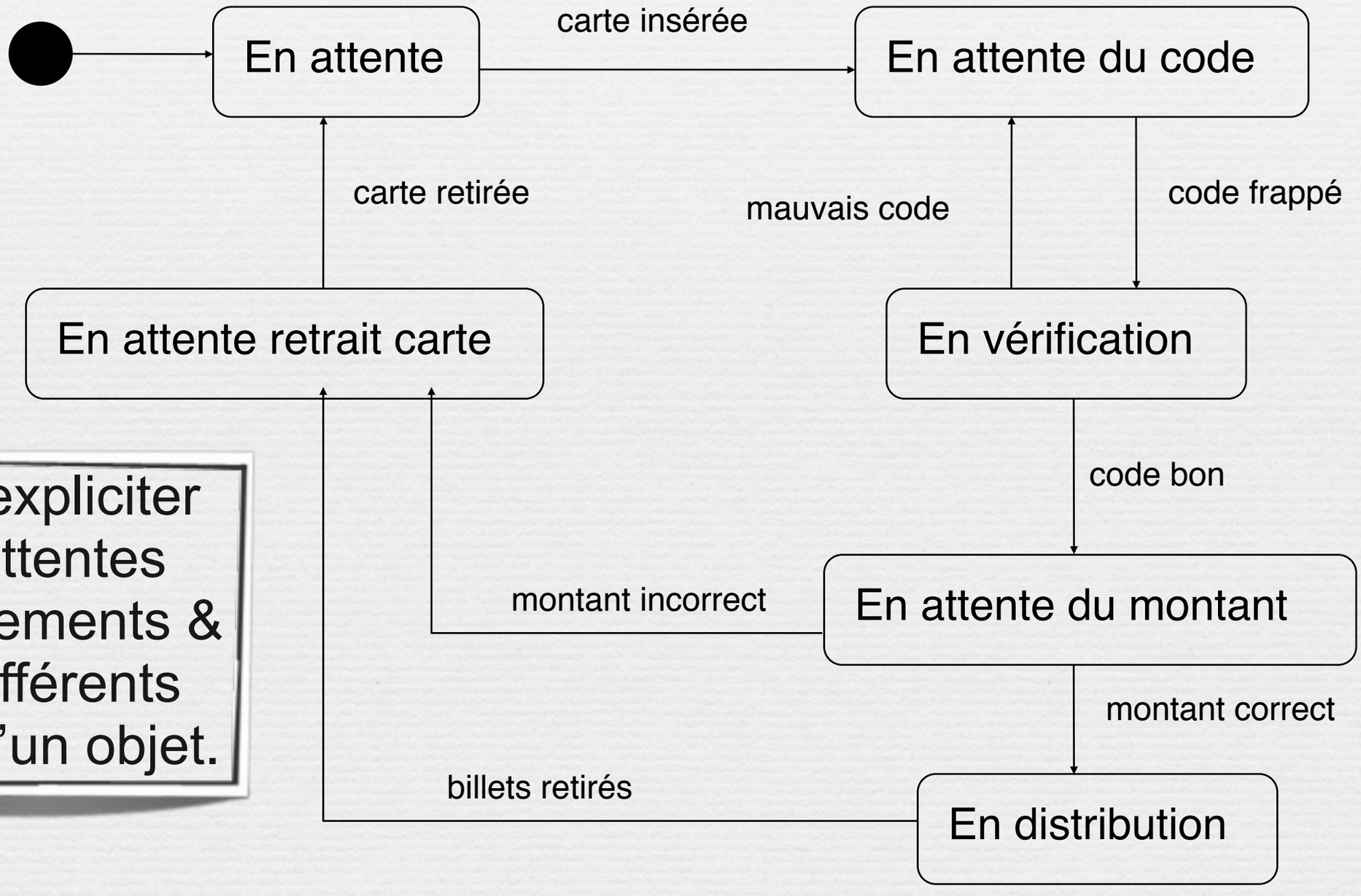
Vue Dynamique

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.

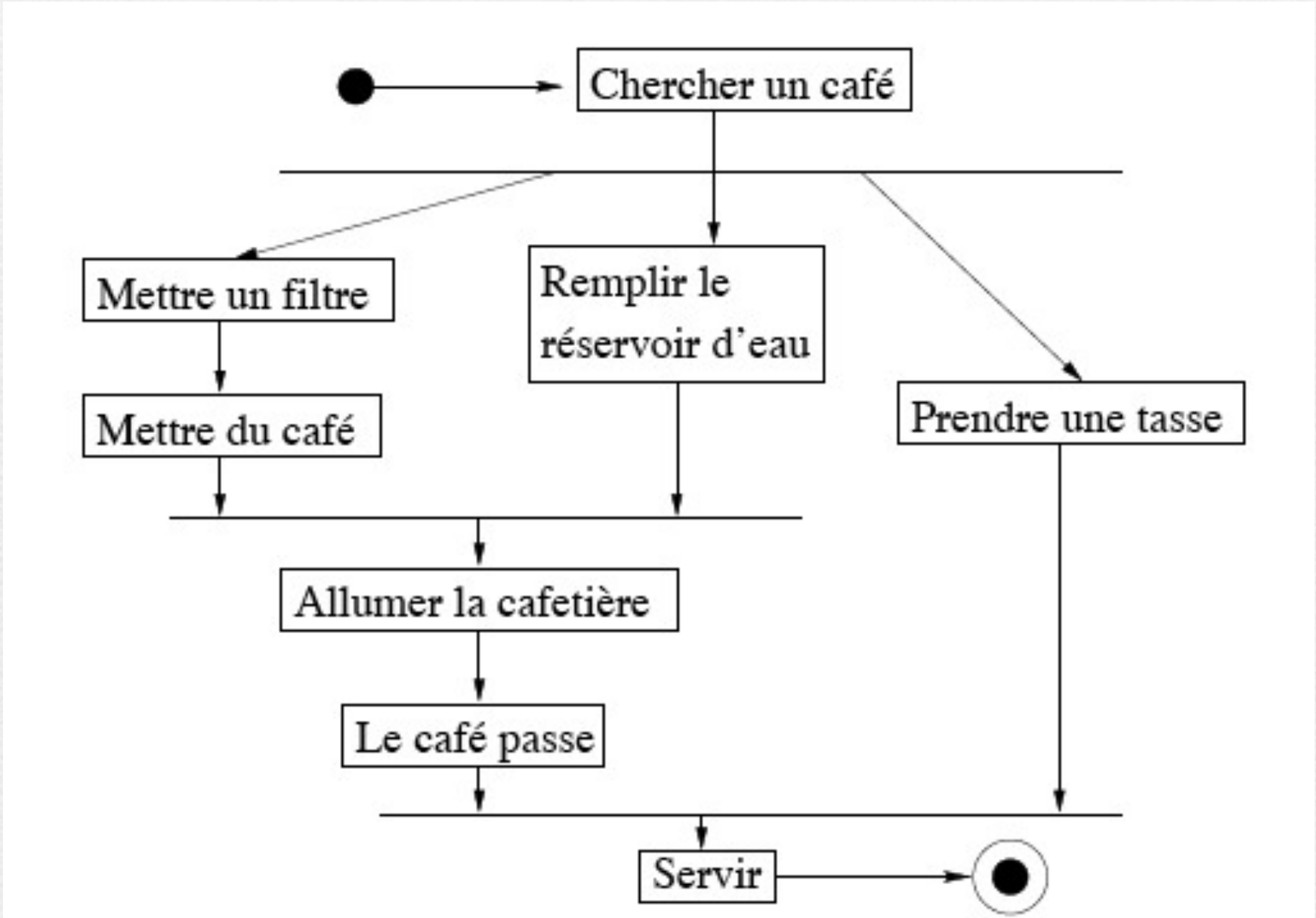
http://wwwdfr.ensta.fr/Cours/docs/IN204/uml.pdf

Qu'est-ce qu'UML ?
Diagrammes d'états



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

Qu'est-ce qu'UML ?
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]

- **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

Différents niveaux de description

Selon l'activité de l'ingénieur, qu'il s'agisse d'analyse, de conception ou d'implémentation, le niveau de détail avec lequel est représenté le diagramme des classes change énormément.

– le point de vue de l'analyse, qui en général se doit d'oublier tout aspect de mise en oeuvre et, en ce sens, est complètement indépendant du logiciel (on n'y parlera pas de structuration des données : tableaux, pointeurs, listes, ...) ;

– le point de vue de la conception, qui cherche à identifier les interfaces, les types des objets, leur comportement externe et la façon interne de les mettre en oeuvre, sans être encore fixé sur un langage ;

– le point de vue de l'implémentation, qui cherche à décrire une classe, ses attributs et ses méthodes en pensant déjà au code qui les implémentera et prend en compte les contraintes matérielles de temps d'exécution, d'architecture, etc.

Dans le cadre d'une analyse, seuls les noms des attributs et les principales méthodes publiques de la classe ont à être mentionnées. Dans le cadre d'une conception et, à plus forte raison, d'une implémentation, la description des classes devra être exhaustive. Mais les différences ne se limitent pas au seul niveau de description. De nombreuses classes spécifiques seront ajoutées lorsque l'on passe de l'analyse à la conception, l'organisation des diagrammes peut évoluer, etc.

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

Bibliographie

Ce cours a été monté en utilisant de nombreux supports dont je remercie chaleureusement ici les auteurs
D'autres références se trouvent sur le site du module.

- Merise: 5ème Partie Dossier "SAM l'Informaticien" du 5 Mars au 18 Mars 2001 par Stéphane Lambert http://www.vediovis.fr/index.php?page=merise5
- Introduction au langage UML, SUPINFO
- De Merise à UML, Nasser Kettani, Dominique Mignet, Eyrolles
- http://www.compucycles.com/nouveausite/articles/Merise/Article_07.htm
- UML-MERISE Etude Comparative, OSITEC-Consultants, 2004-2005
- Modélisation Orientée objet, M.Grimaldi – janvier 2010