Bases de la conception orientée objet

1e année IUT - Semestre 2

Ivan Logre
à partir des supports de
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

- **Connaître la modélisation UML**
  - Savoir lire des modèles ; savoir les construire
  - Faire le lien entre un modèle et le code qui pourrait correspondre.

Objectifs du cours

- **Connaître la modélisation UML**
  - Savoir lire des modèles ; savoir les construire
  - Faire le lien entre un modèle et le code qui pourrait correspondre.

- **Savoir analyser un problème pour ensuite pouvoir l'implémenter**
  - En répondant aux besoins des utilisateurs
  - En assurant la qualité du logiciel produit (performance, utilisabilité, sécurité, maintenabilité, ...)

Notation

- Notes de TD (participation, effort, progrès) [0.75]
- Contrôle intermédiaire (QCM ou étude de cas) [0.75]
- Projet (Note individuelle au sein d'un groupe) [1]

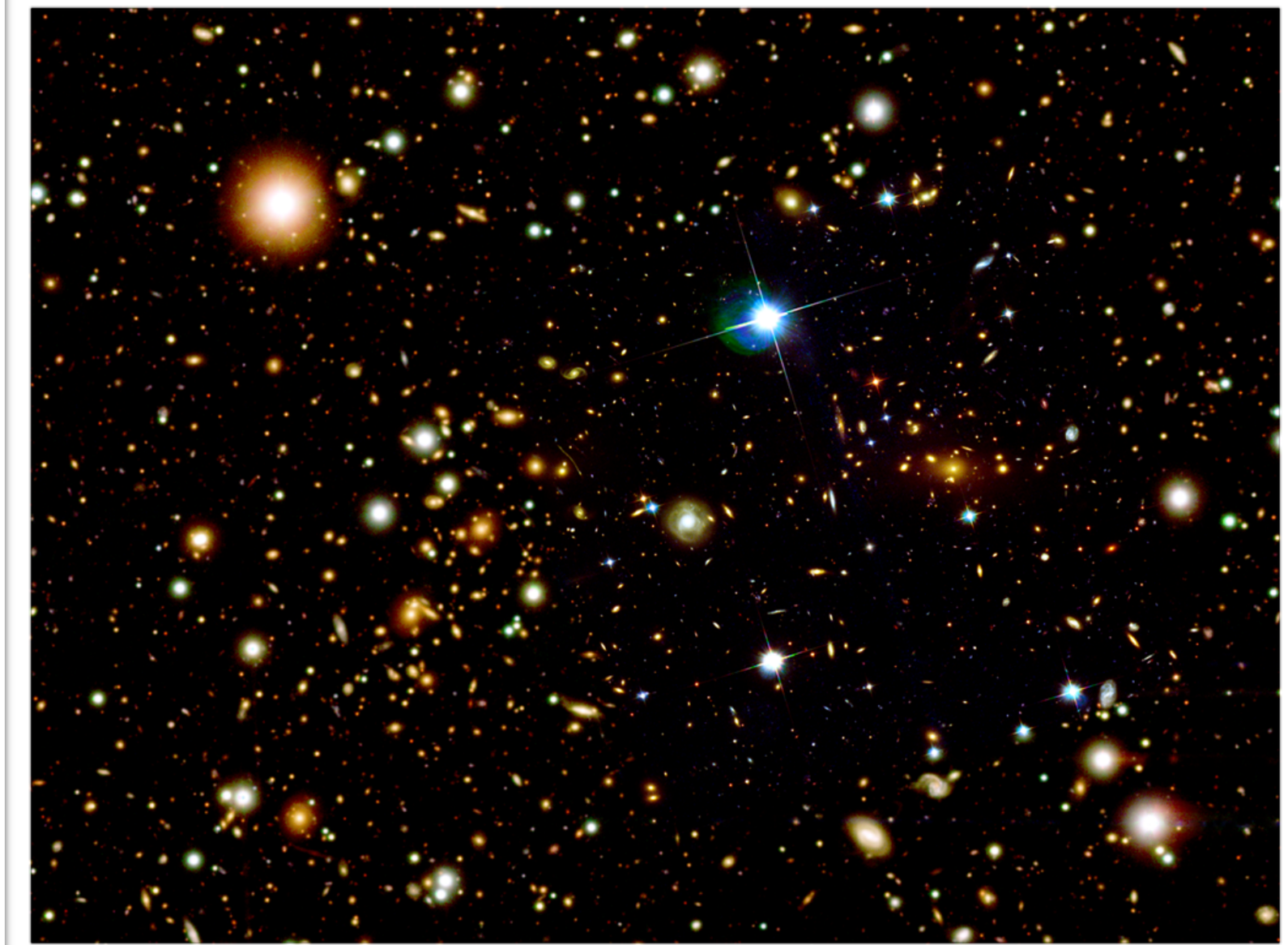
Notation

- Notes de TD (participation, effort, progrès) [0.75]
- Contrôle intermédiaire (QCM ou étude de cas) [0.75]
- Projet (Note individuelle au sein d'un groupe) [1]

Soyez Rigoureux.

Respectez: dates,notations,consignes.

[1] Motivations



L’univers du logiciel

Où se trouve le logiciel ?

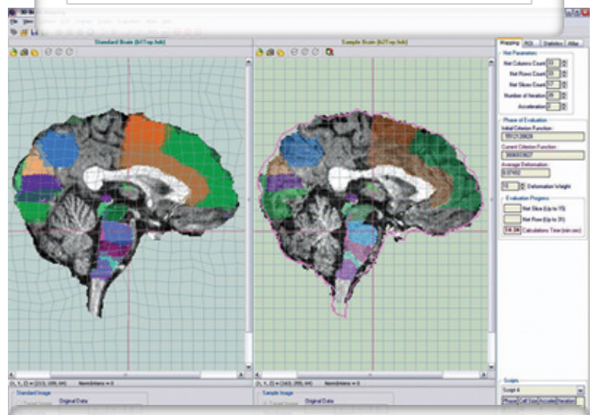
Où se trouve le logiciel ?

Finances

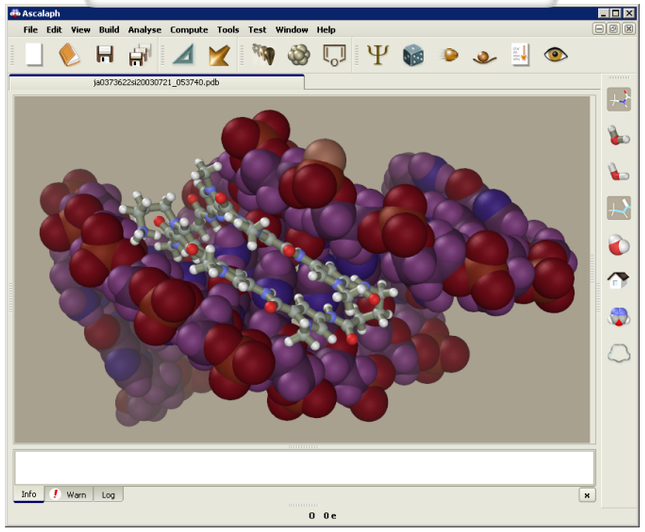
Transports

Télécommunication

Médecine



Sciences



Domotique

Industrie

Web

Fouille de données

Nouvelles interactions

Musique, sport, arts...

Mise en situation

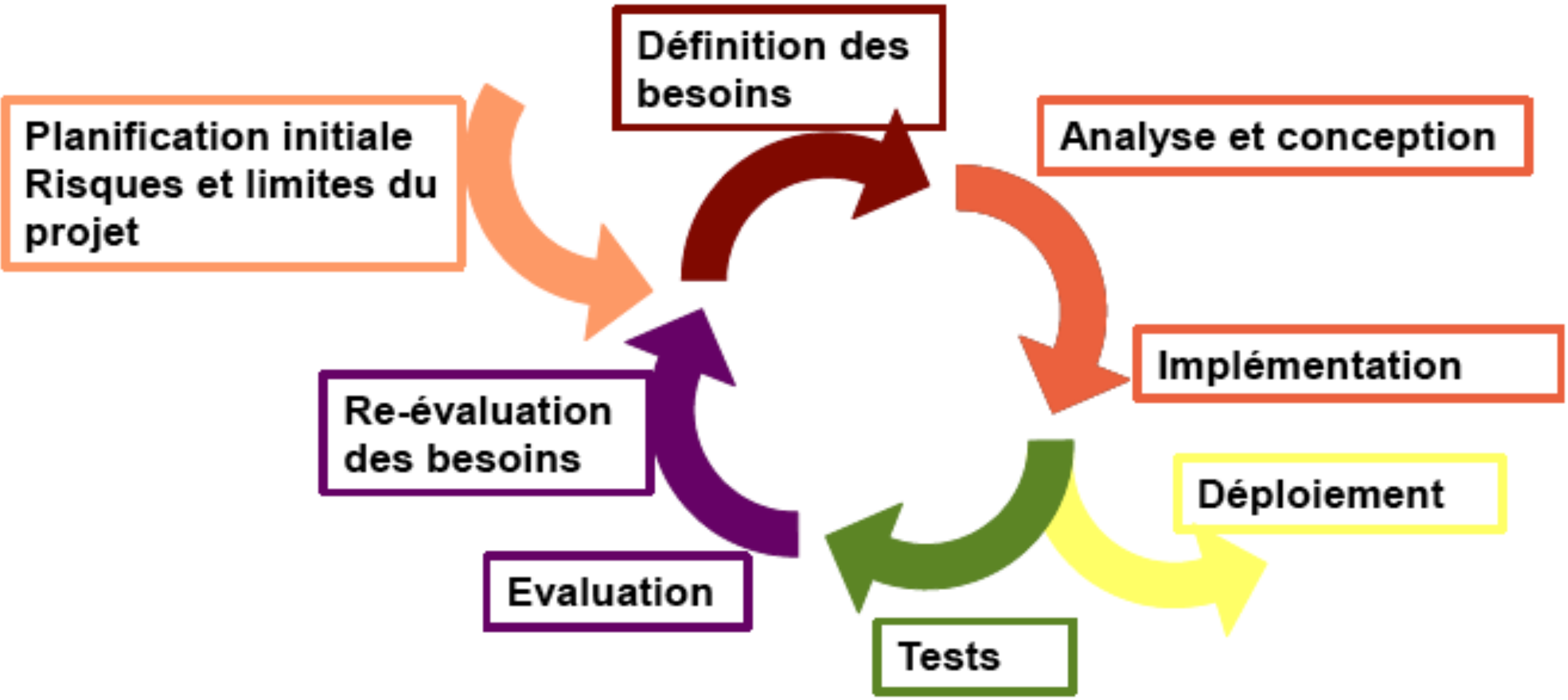
Vous devez réaliser une application logicielle pour visualiser les données météo dans un cockpit, que faîtes-vous?

Mise en situation

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

C'est le moment de participer

Activités du développement logiciel

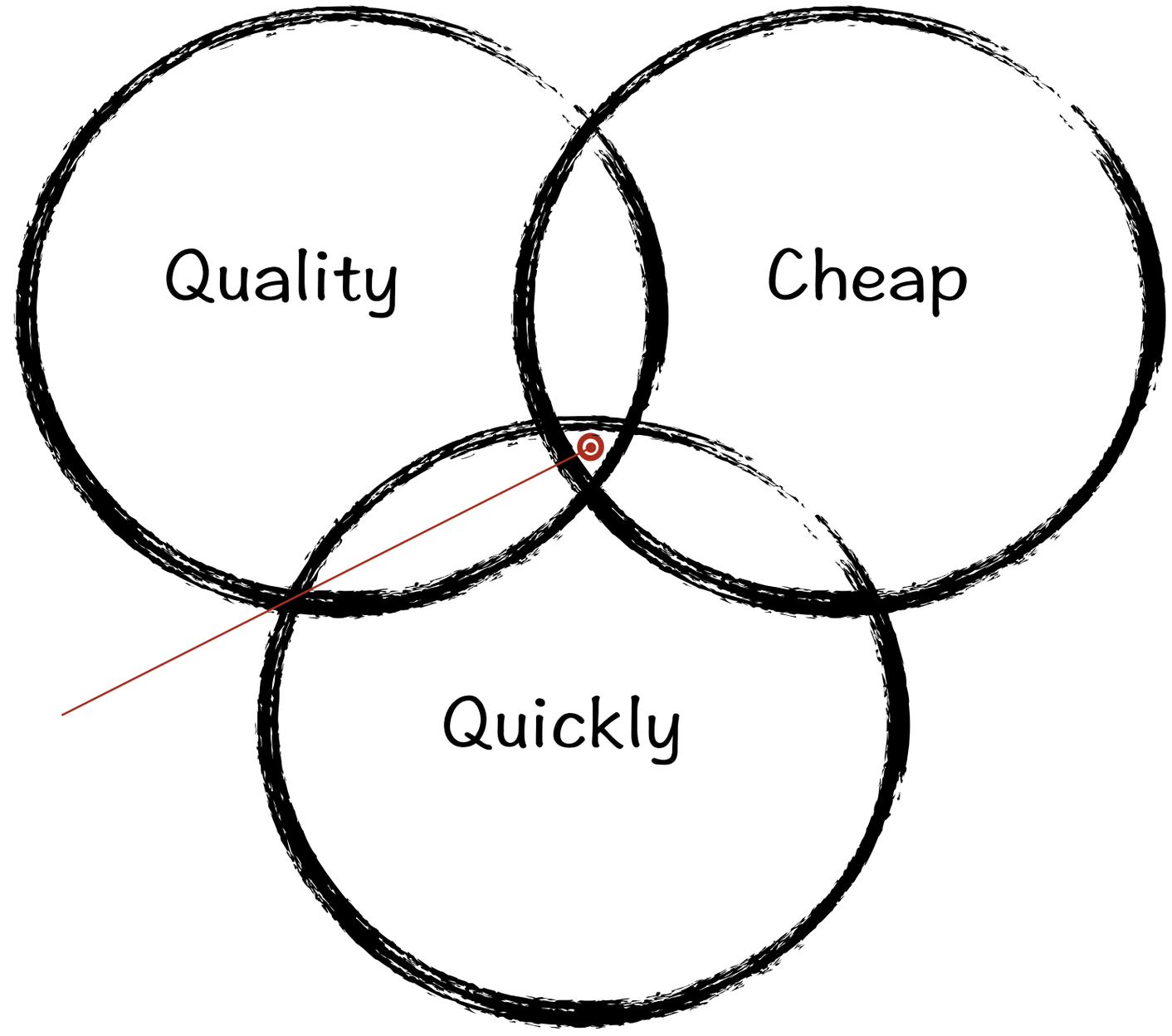


Extrait de Bonnes Pratiques de développement
ENVOL_2010: 27 Septembre
Véronique Baudin
Violaine Louvet

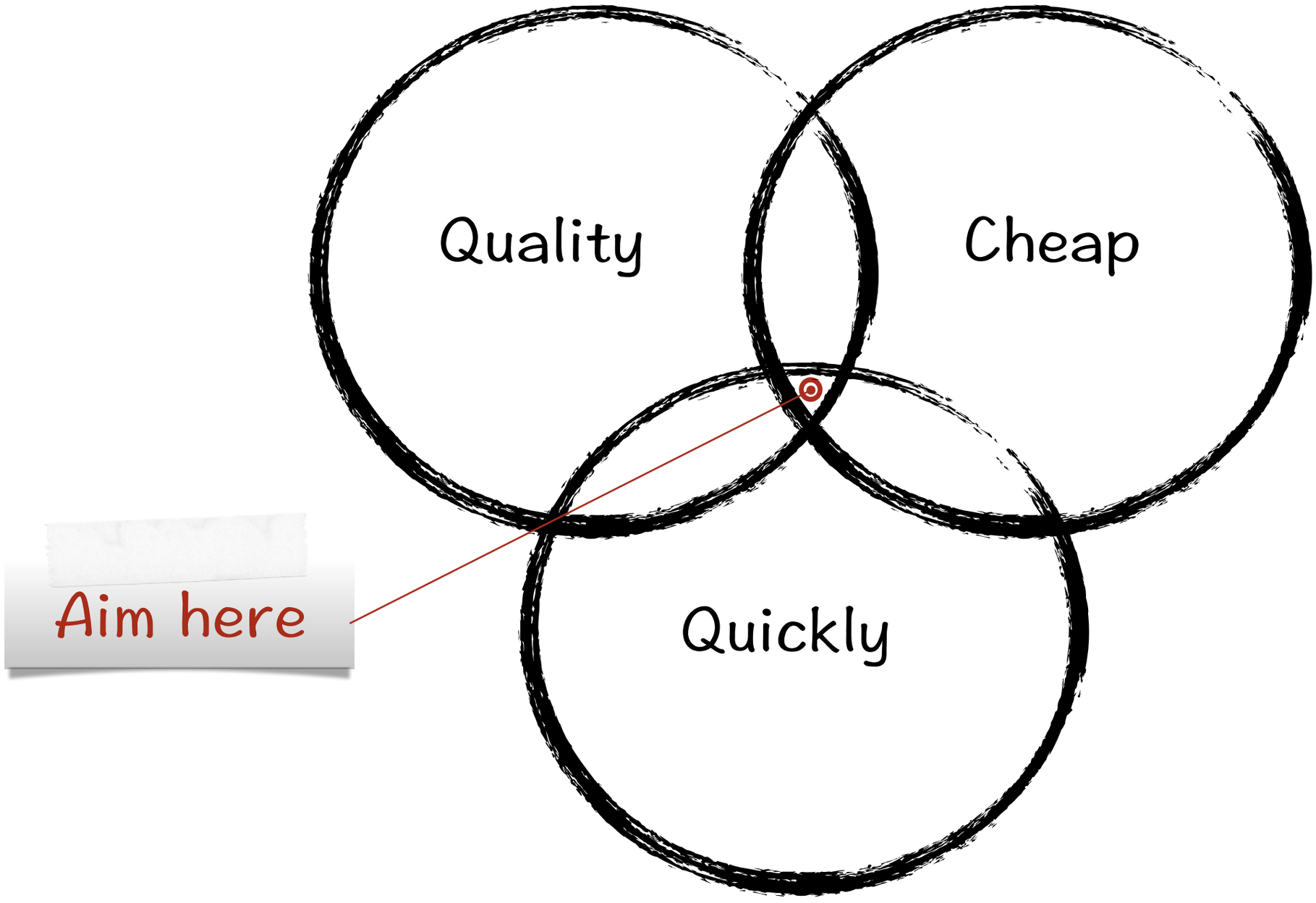
Monteriez-vous dans l'avion pour lequel vous avez développé le système de pilote automatique ?

Monteriez-vous dans l'avion pour lequel vous avez développé le système de pilote automatique ?

Quelle Qualité Logicielle ?



Quelle Qualité Logicielle ?



[2] UML, un peu d’histoire

Fin des années 90

(Oui, j'y étais)

Des méthodes !

(a.k.a. "quelque soit votre problème, on a la solution")

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

Retour d'expérience : Meh.
Et si on définissait d'abord un langage de modélisation standard?

Et un langage unique, un !

UNIFIED MODELING LANGUAGE

Un cocktail de notations éprouvées.

(...enfin presque toutes, p. ex. RdP, SADT/IDEF0, DFD, etc.)

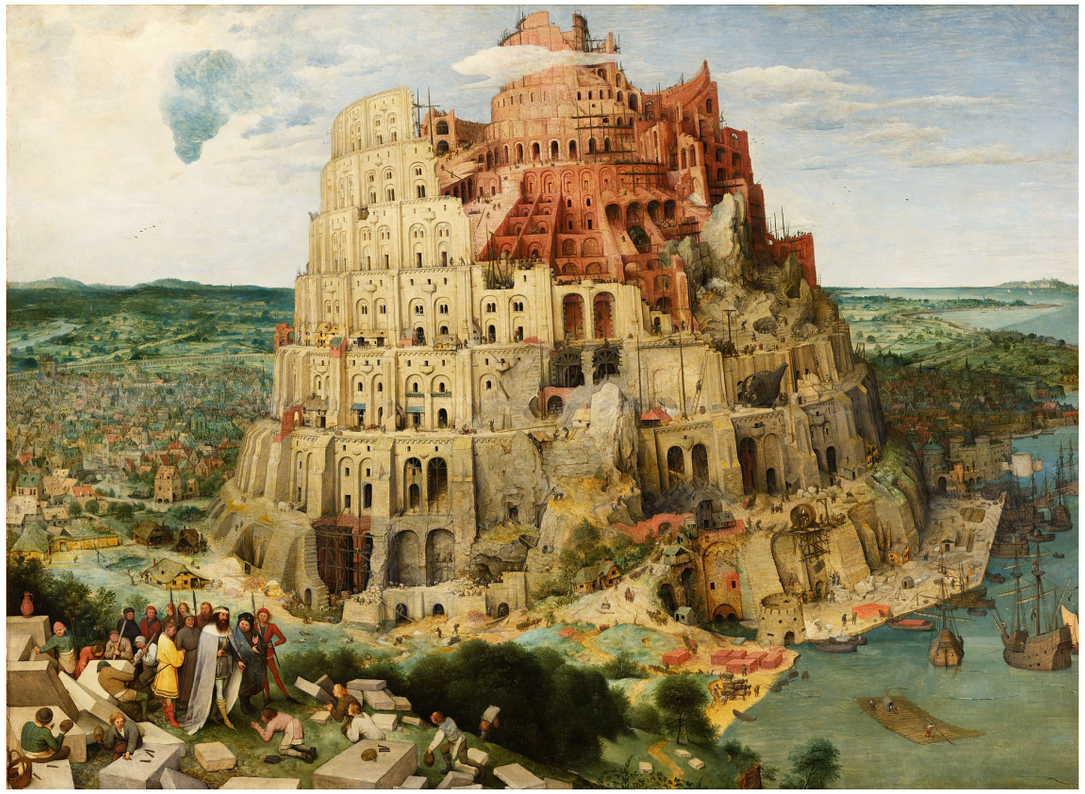
- Auteurs : Grady Booch, Ivar Jacobson, James Rumbaugh.
- Standardisation OMG en 1997
- Promoteurs :
  - Rational Software, Oracle
  - HP, Microsoft, IBM

OBJECT MANAGEMENT GROUP

http://www.omg.org/spec/UML/2.5/PDF/

Objectif : Tour de Babel

(Résultat : tour de Pise?)



Standard d'UML 2.5 (2015)

794 pages

http://www.omg.org/spec/UML/2.5/PDF/

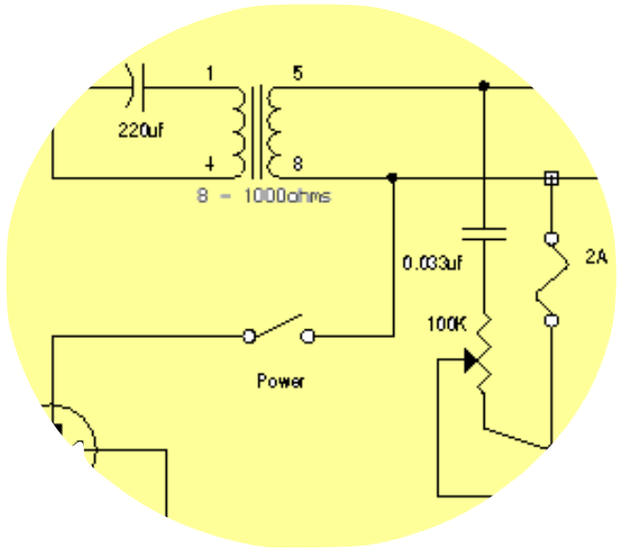
96% des développeurs le connaissent
56% l'utilisent en développement

(en 2005, sur 500 répondants)

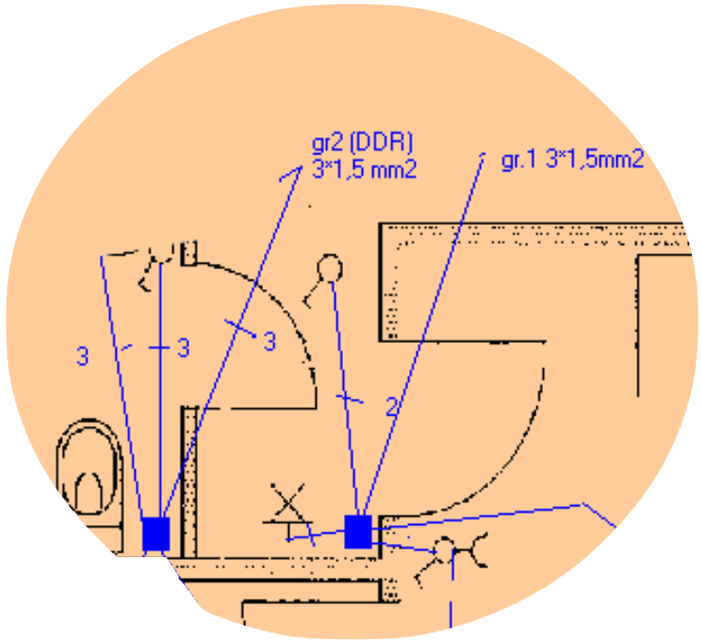
http://www.prweb.com/releases/2005/04/prweb231386.htm

À chacun son métier
À chacun son langage

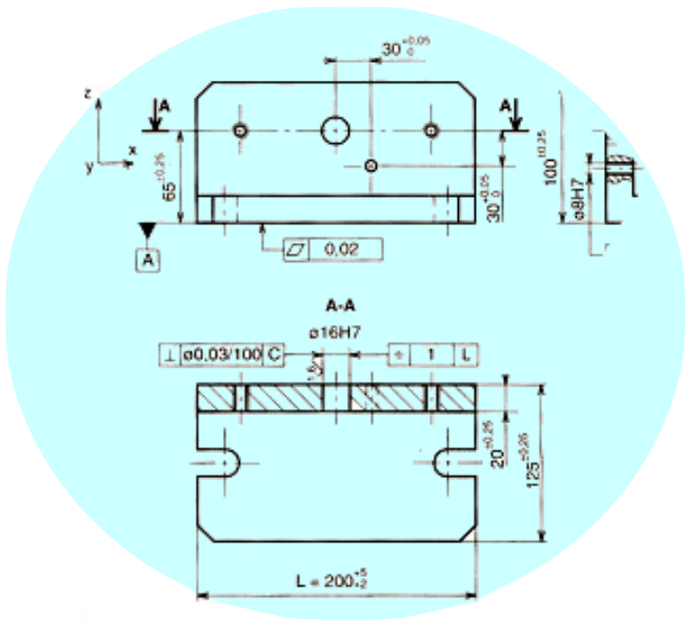
Ingénierie Electronique



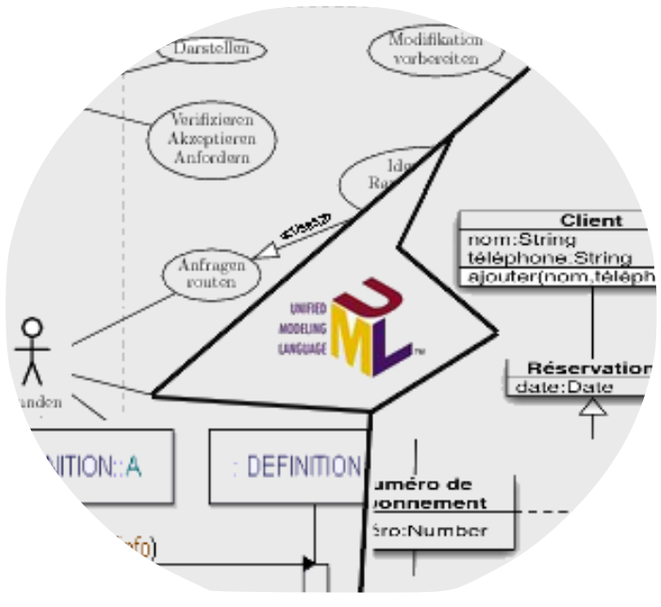
Ingénierie du Bâtiment



Ingénierie Mécanique



Ingénierie Logicielle



=> un support à la **modélisation**

Qu'est ce qu'un modèle ?

C'est le moment de participer

Qu'est ce qu'un modèle ?

Une **simplification** de la réalité, afin de

Visualiser le système

Spécifier la structure
le comportement

Aider et guider
la construction du système

Documenter
les décision

Qu'est ce qu'un modèle ?

Une **simplification** de la réalité, afin de

Visualiser le système

Spécifier la structure
le comportement

Aider et guider
la construction du système

Documenter
les décision

Résumé : Qu'est ce qu'UML ?

UML est un langage
UML n'est pas une méthodologie

UML permet
une description graphique et textuelle
de l'architecture et du comportement

UML supporte:
✓ la visualisation,
✓ la spécification,
✓ la construction,
✓ la documentation.

(Cerise sur le gâteau, on peut même générer du code)

Points forts d'UML ?

Un langage normalisé:
précision
stabilité
interopérabilité

Support de communication
polyvalent et souple
facilite la compréhension de représentations abstraites complexes

Points faibles d'UML ?

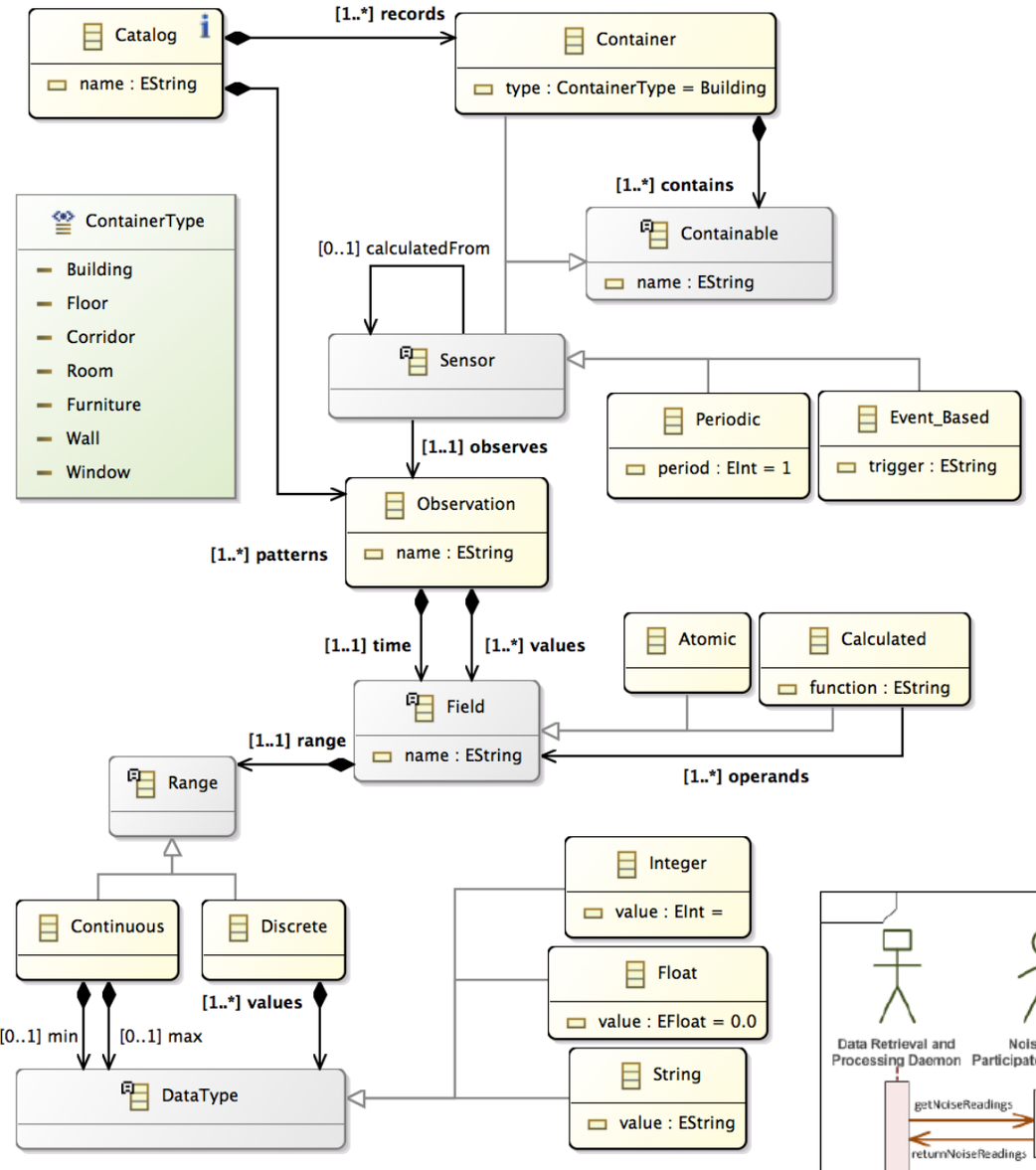
"Unified" => Complexité => Apprentissage

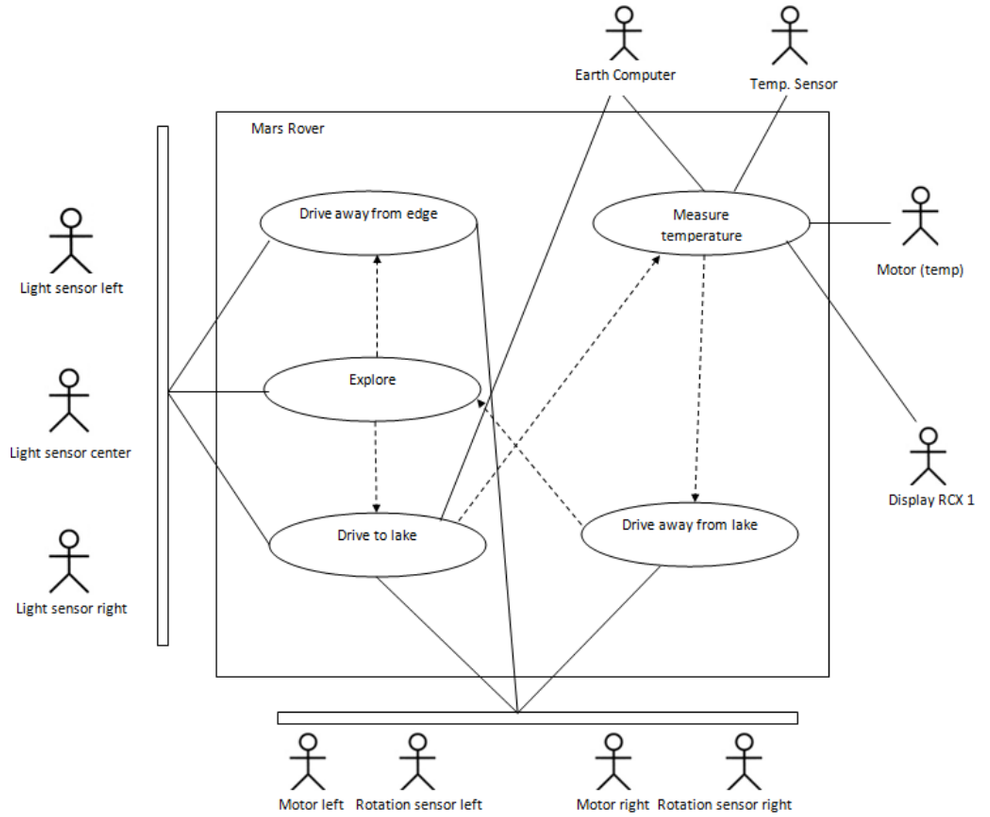
(794 pages pour définir un langage)

"Language" =/= méthodologie

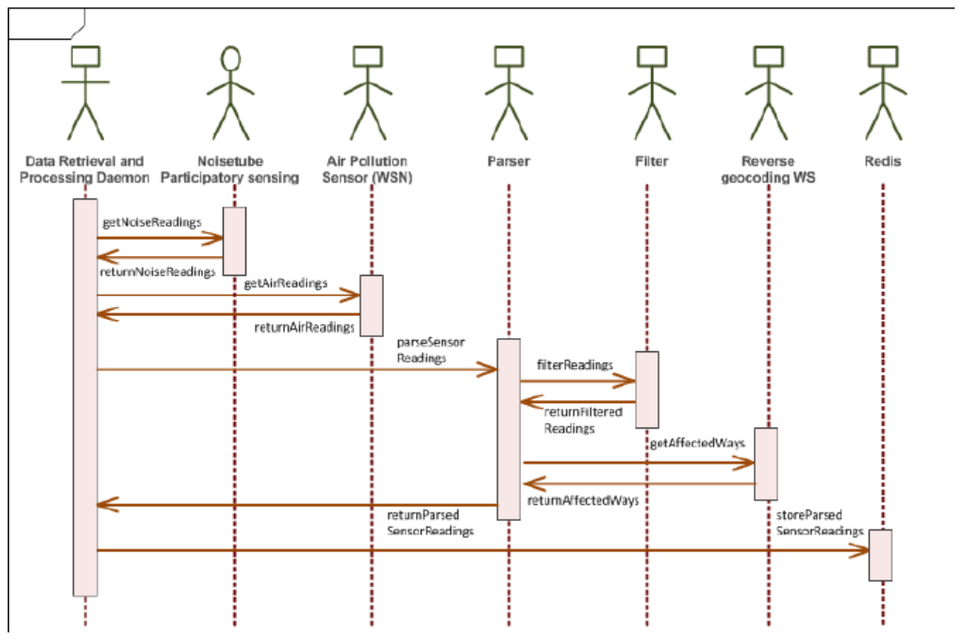
le processus de développement reste une difficulté

[3] UML, un survol

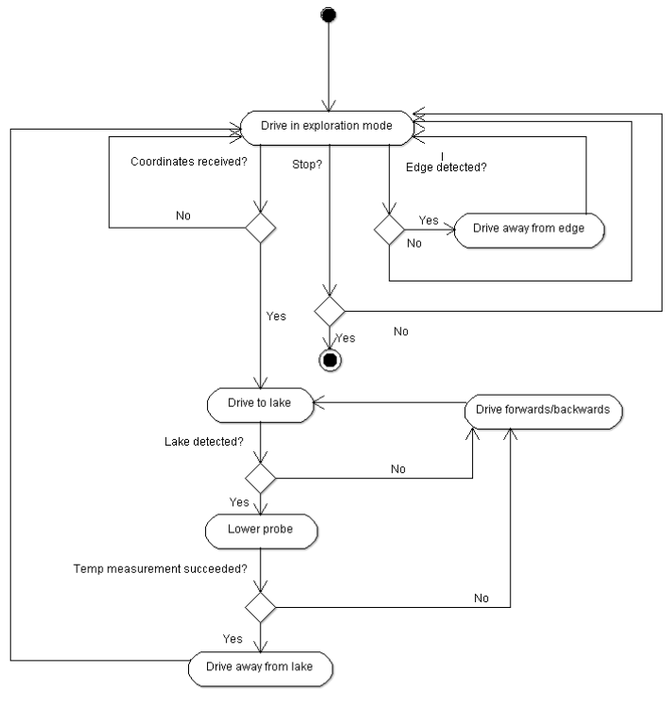




https://embeddedmotioncontrol.wordpress.com/2011/05/18/its-uml-time/



Vivek Nallur et al, Smart Route Planning Using Open Data and Participatory Sensing (2015)



Vue fonctionnelle

Interactions entre les acteurs et le système

Exemples :

Objectifs à atteindre (**cas d'utilisation**)

Scénarios d'intéraction typiques (**diag. de séquence**)

Diagramme des cas d'utilisation

Montre ce que fait le système et qui l'utilise

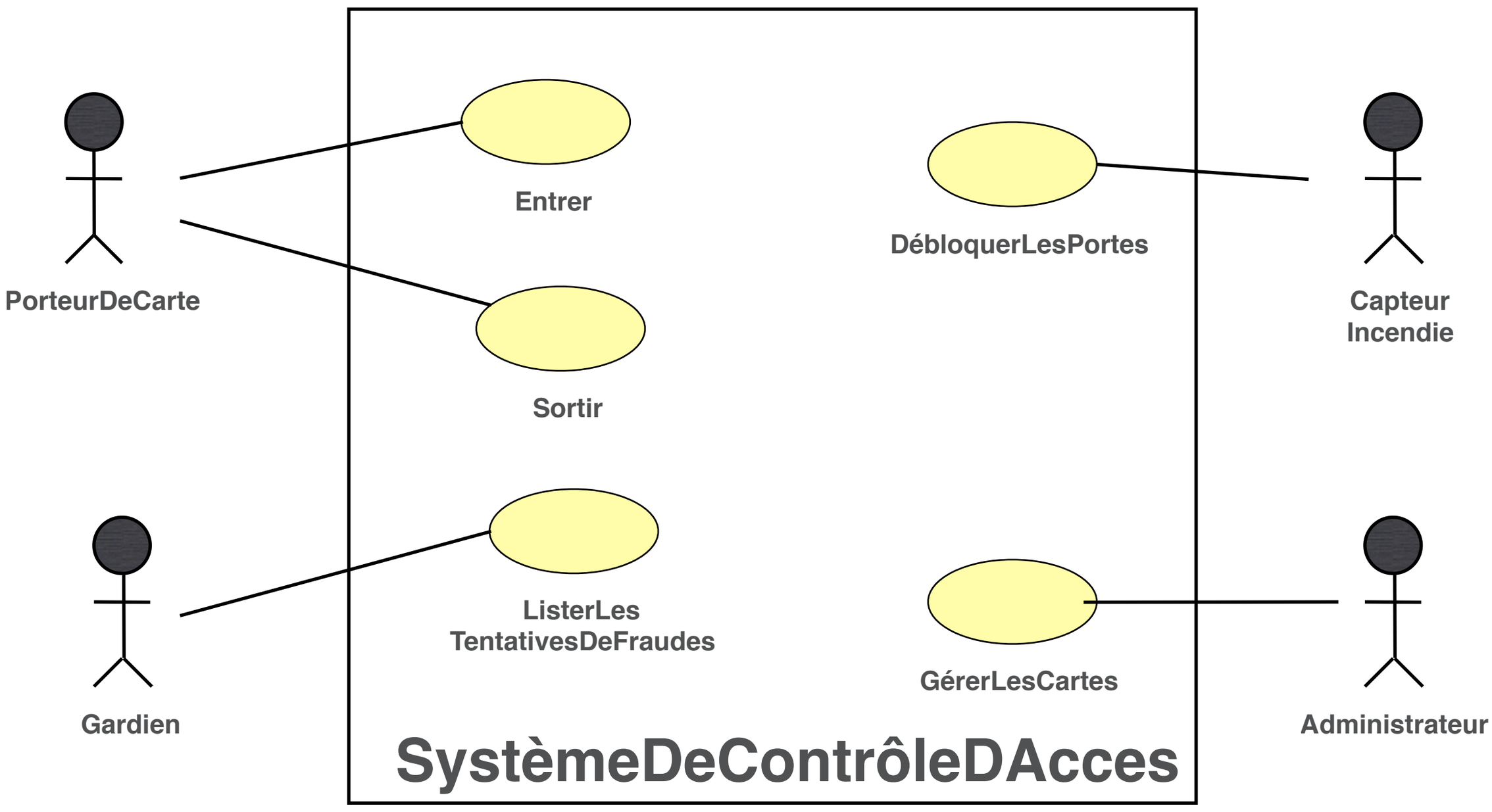


Diagramme de séquence

Montre les flux de communication au sein d'un scénario
donne un aspect temporel

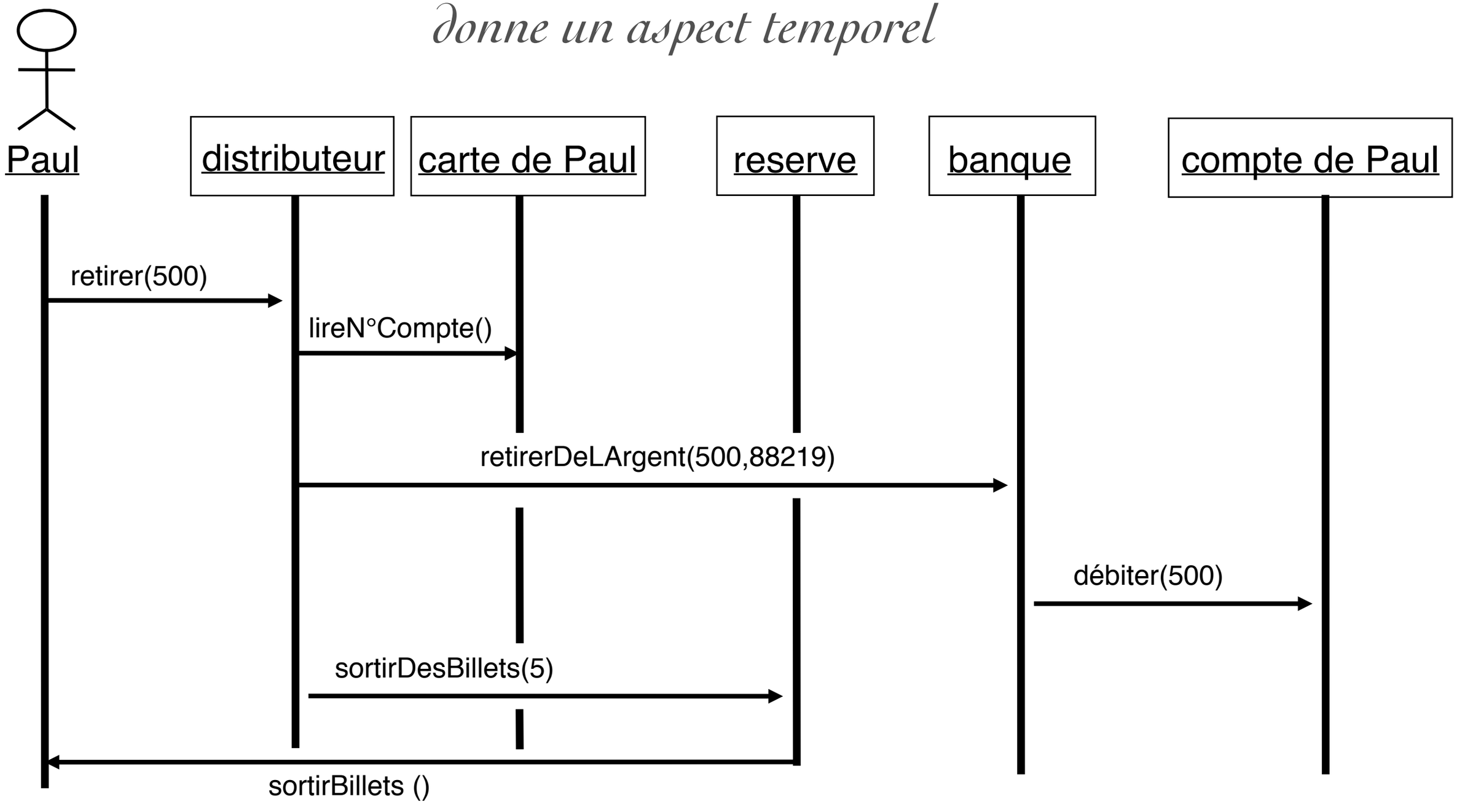
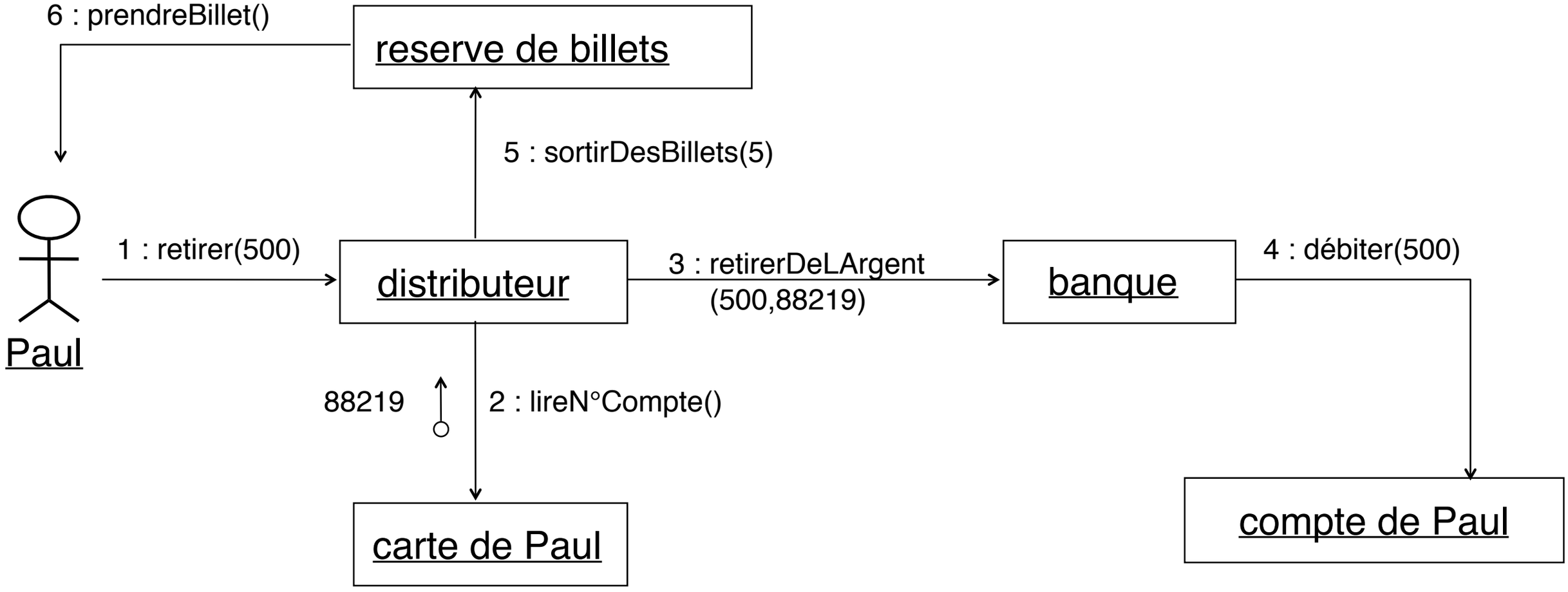


Diagramme de collaboration

Montre les flux de communication au sein d'un scénario en se centrant sur les objets impliqués



Non étudié cette année

Vue structurelle

Identification des constituants du système (*ie.* objets/composants)

<u>Exemples :</u>

Identifier pour chaque classe ses attributs, opérations, méthodes, leurs liens… (**diag. de classe**)

Regrouper les classes fortement liées en composants autonomes (**diag. de package**)

Diagramme de classe

Montre les classes, leurs constituants et leurs relations

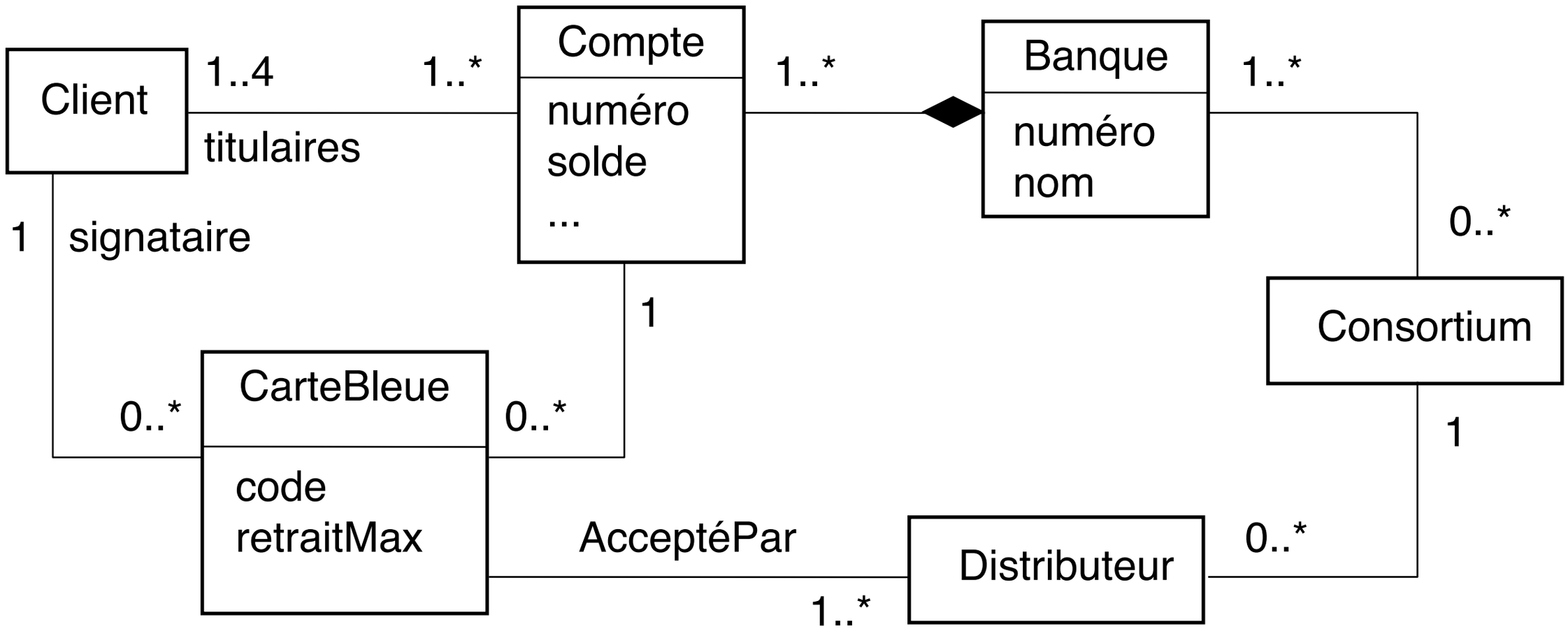
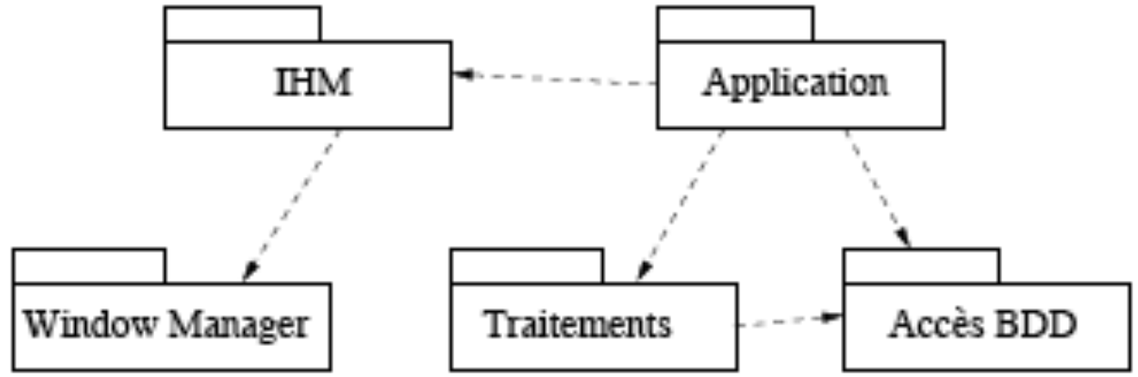


Diagramme de packages

Montre les regroupements de classes et leurs relations

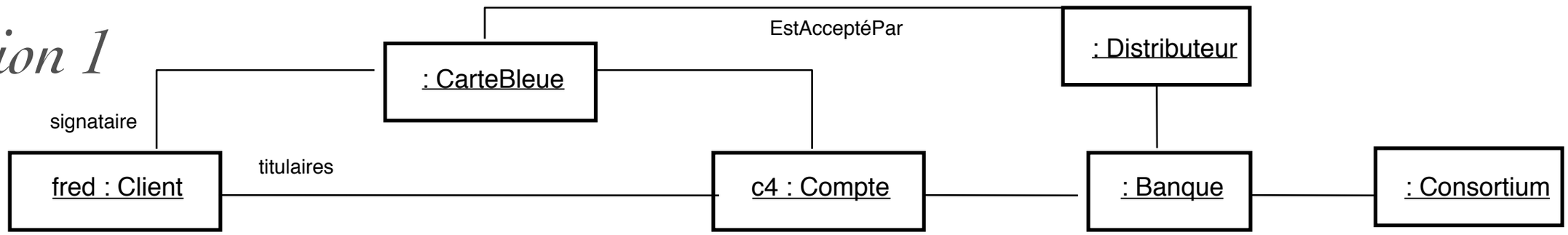


Regrouper entre elles des classes liées les unes aux autres
Faciliter la maintenance et l'évolution du projet
Rendre plus indépendantes les différentes parties d'un logiciel

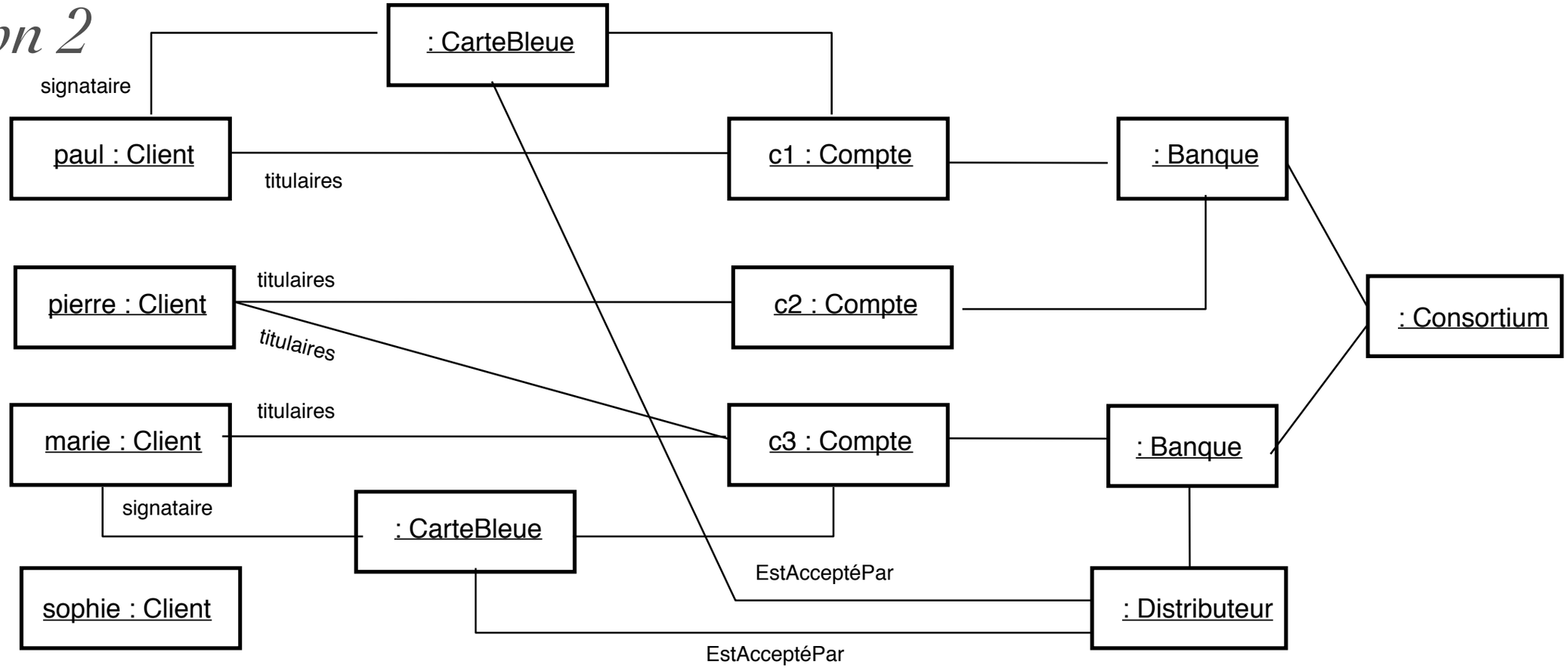
Diagramme d'objet

Montre les instances et leurs liens à l'exécution

Exécution 1



Exécution 2



Vue dynamique

Description de l'évolution des objets au long de leur cycle de vie

<u>Exemples :</u>

Changement d'état de la naissance à la mort d'un objet du fait des interactions (**diag. d'état**)

Parallélisme d'activité, synchronisation lors de l'exécution pour assurer la cohérence (**diag. d'activité**)

Diagramme d'états

Montre les attentes d'évènements et les différents états d'un objet

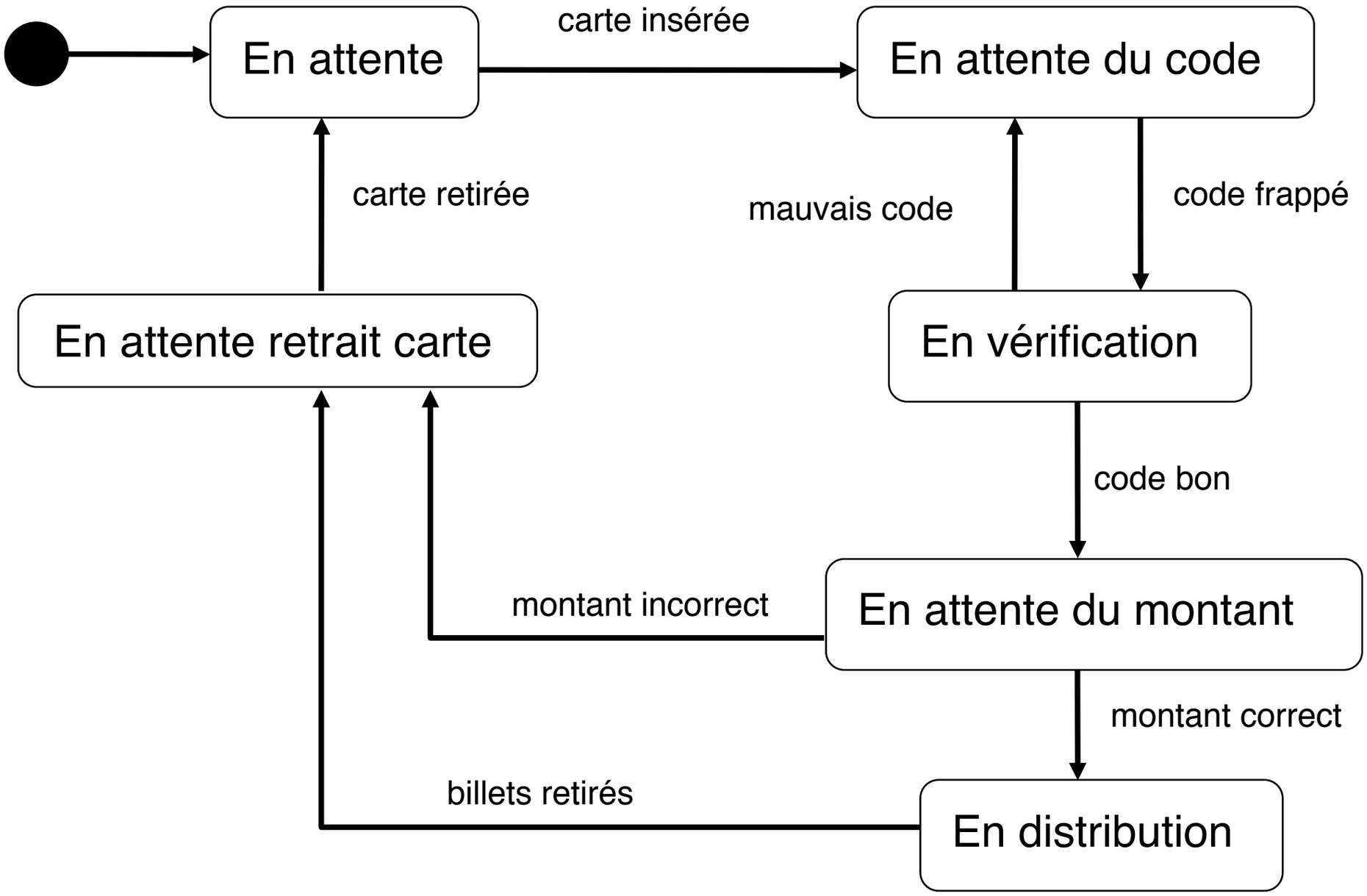
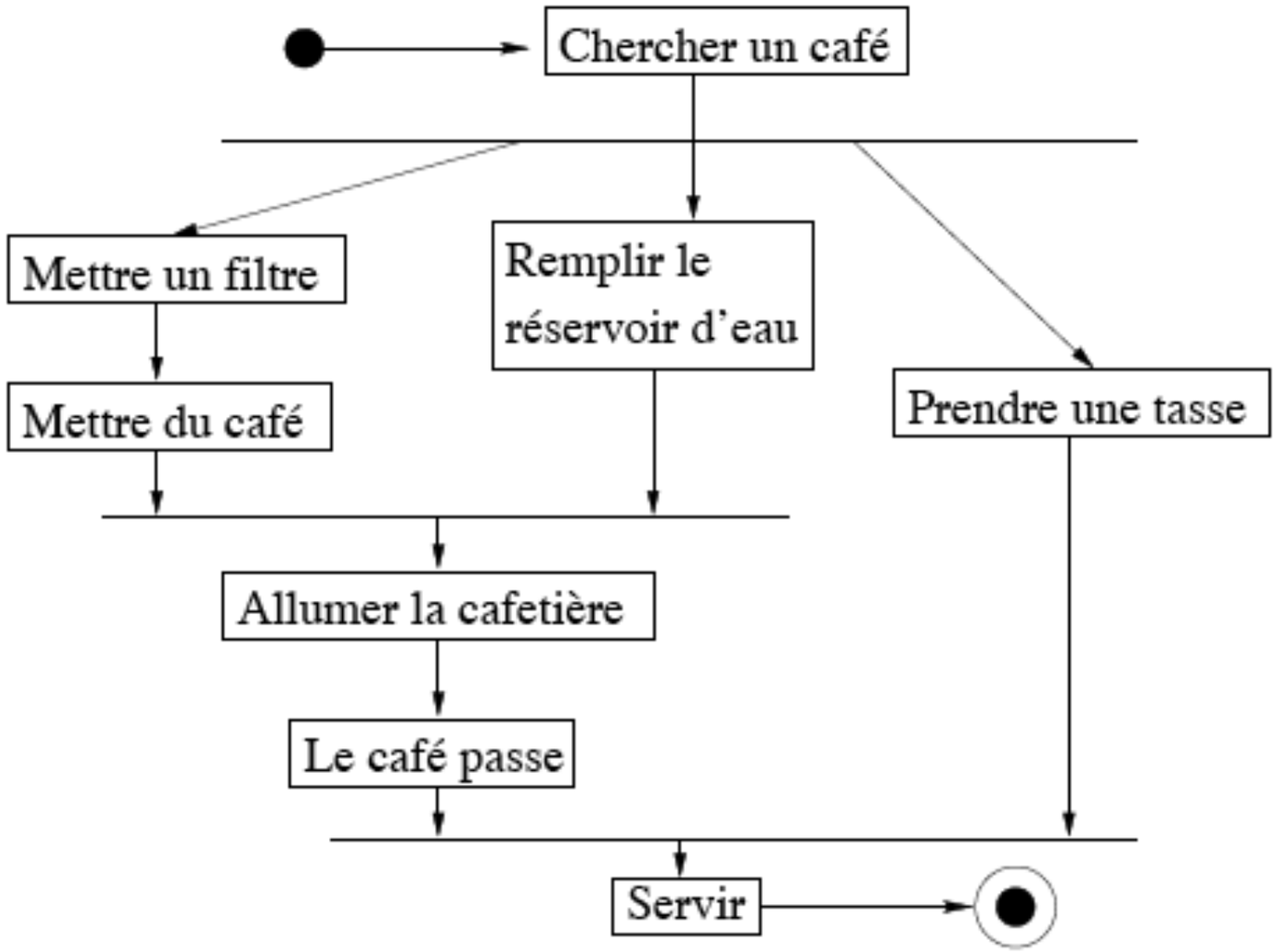
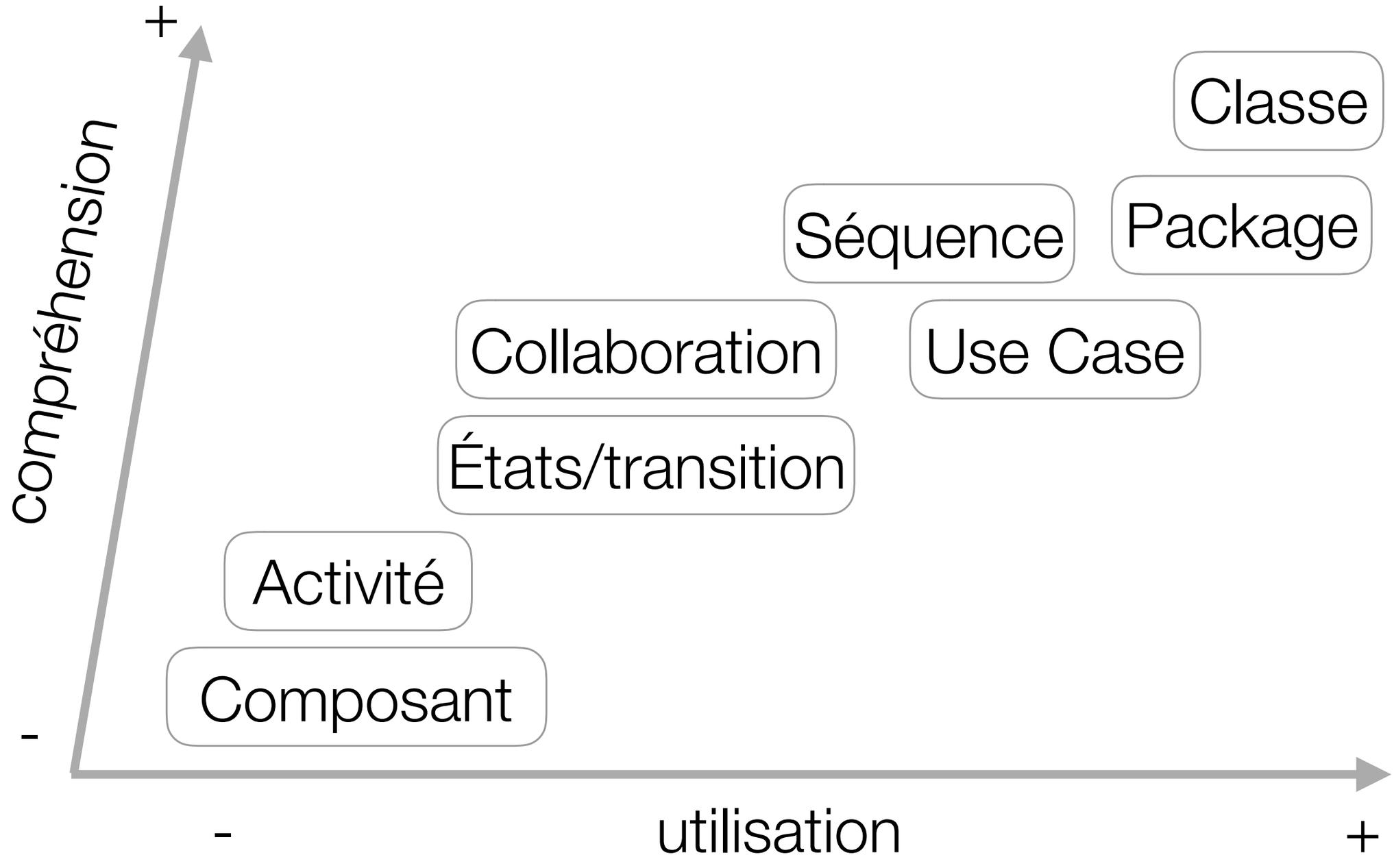


Diagramme d'activité

Explicite des besoins de synchronisation de processus au sein d'un scénario



UML dans l'industrie



Modes d'utilisation d'UML

Mode Esquisse (*sketch*)

Informel, incomplet, manuel
Support de communication
pour concevoir des parties critiques

Mode Plan (*blue print*)

Détaillé, formel, outillé
Génération de squelettes de code
Complétion du code par un développeur

Mode Langage de Prog

Complet, outillé, **exécutable**
Pas vraiment disponible actuellement!

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.

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