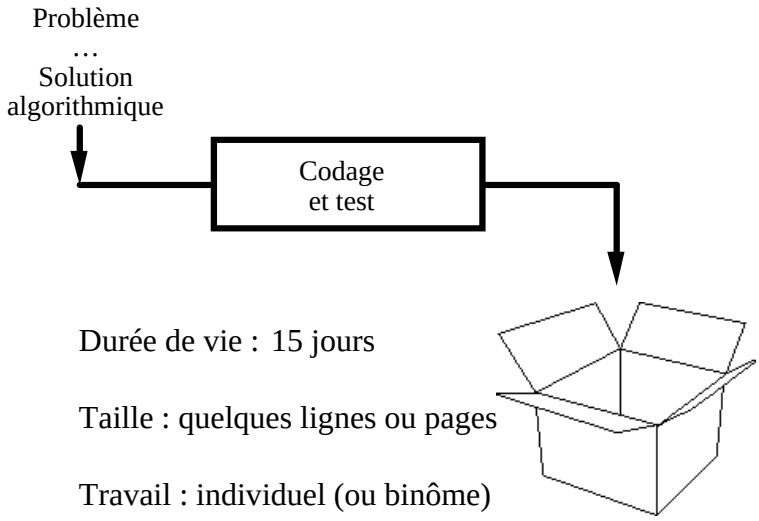
# Penser Objet

cours Programmation Orientée Objet (4/4)

Place de l'approche objet dans le développement des logiciels

- Problématique de projets informatiques
- 3 étapes importantes
- Démarche globale (cycle de vie)
- Support idéal d'aujourd'hui : UML
- Exemple : GAB

## Réalisations de TP

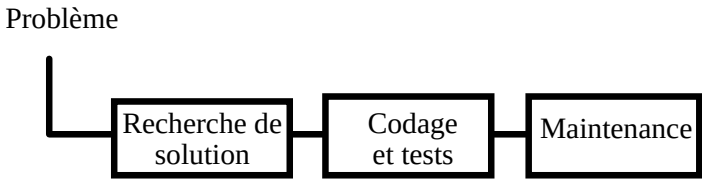



Durée de vie : 15 jours

Taille : quelques lignes ou pages

Travail : individuel (ou binôme)

## Réalisations des projets informatiques industriels




Durée de vie : plusieurs années
Taille : des milliers de lignes
Travail : collectif



## Problématique de projets informatiques

- Gérer un projet informatique :
  - découvrir les réels objectifs visés par le client
  - concevoir de façon consistante
  - réaliser et tester
  - valider globalement
  - faire recetter par le client
- Assurer la qualité et la fiabilité
- Permettre l'évolutivité
- Assurer la maintenance à court et à long terme (curative, évolutive, adaptative)

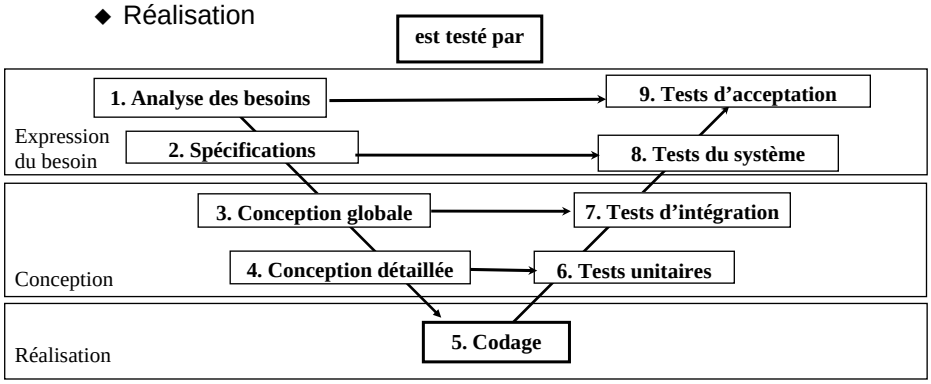
découvrir les réels objectifs visés par le client :
- Cahier des charges : par le client
- Spécifications : par le développeur

concevoir de façon consistante :
- Conception d'architecture
- Conception détaillée

réaliser et tester :
- Programmation
- Test unitaire

## Démarche globale (cycle de vie)

- 3 étapes importantes :
  - Expression du besoin,
  - Conception,
  - Réalisation

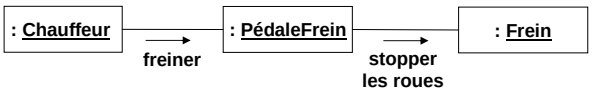





## Penser objet : Des fonctionnalités aux Objets

Fonctionnalité : Freiner

Les objets participants à la mise en place de cette fonctionnalité




Un message entraîne l'activation d'un ou de plusieurs services de l'objet

## Penser objet

- Faire une analyse portant sur les « Etres » du domaine d 'application (objets)
- Identifier à la fois leurs caractéristiques et leurs services
- Avoir une approche recherchant des modèles (représentants) donc classes
- Faire passer au second plan le fonctionnement global
- Identifier les liens statiques (relationnels) entre les objets (pour constituer un référentiel)
- Identifier des besoins d'utilisation de services proposés par d'autres classes



## UML (Unified Modeling Language)

- Normalisé en 1997 par l'OMG (Object Management Group) qui est aussi à l'origine de CORBA
- Version courante industriellement utilisée : 1.5
- Version 2.0 juste normalisée : bien plus complexe
- UML:
  - un langage graphique (supporté par un méta-modèle)
  - un ensemble de vues (=diagrammes) modélisant les aspects statiques et dynamiques
- Il ne propose pas UNE méthodologie d'analyse et de conception, mais peut en supporter plusieurs !

⇨ Méthodes : Processus Unifié (RUP – Rational Unified Process)

Model-Driven Architecture (MDA) – supportée par OMG

## Composants du formalisme UML

- **Diagramme de classes**
- **Diagramme d'objets**
- **Diagramme de cas d'utilisation**
- **Diagramme d'états**
- **Diagramme de séquences**
- **Diagramme d'activités**
- **Diagramme de collaboration**

- **Diagramme de composants**
- **Diagramme de déploiement**



## Exemple : Guichet Automatique de Banque (1/3)

- Le client pourra accéder grâce au GAB, à un compte courant et un compte épargne. Les transactions autorisées sont les dépôts et les retraits en liquide ; les retraits seront toujours des multiples de 20 € ; avec un maximum de 300 € par jour.

- L'identification des clients doit aller au delà de la simple possession de la carte d'accès au GAB. Le système vérifiera la validité d'accès physique, et de l'identification du client, de la validité des comptes accessibles à travers le GAB, et que le client n'a pas fait de déclaration signalant une mise en danger de la sécurité du système (perte ou vol dans le cas d'une carte d'accès).

## GAB (2/3)

- Le GAB doit s'assurer que le compte du client contient suffisamment de fonds pour couvrir les demandes de retrait, et doit communiquer à l'ordinateur central les débits (retraits) et crédits (dépôts).
- Une transaction est un dépôt ou un retrait ; le client reçoit un reçu pour chaque transaction. Le système doit garantir la complétude des transactions : si une transaction est en cours, le GAB affichera un message d'avertissement et annulera la transaction.
- Le système doit annuler une transaction en cours et annuler l'autorisation d'accès si le client ne répond pas (correctement) à une demande du système dans un délai imparti raisonnable et/ou un nombre d'essais limite. Le système doit aussi répondre aux requêtes clients dans un temps raisonnable. La vérification d'erreurs et la sécurité physique font aussi partie du problème.

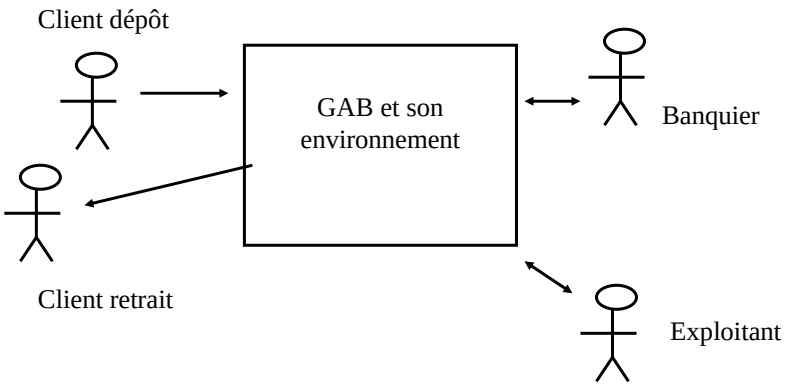


## GAB (3/3)

- Le GAB doit assurer un fonctionnement 24h sur 24, tous les jours, et doit être opérationnel 90% du temps. Pour éviter des mises hors services trop longues, les réparations entreprises sur le GAB ne doivent pas dépasser 2 heures (temps de réparation moyen).

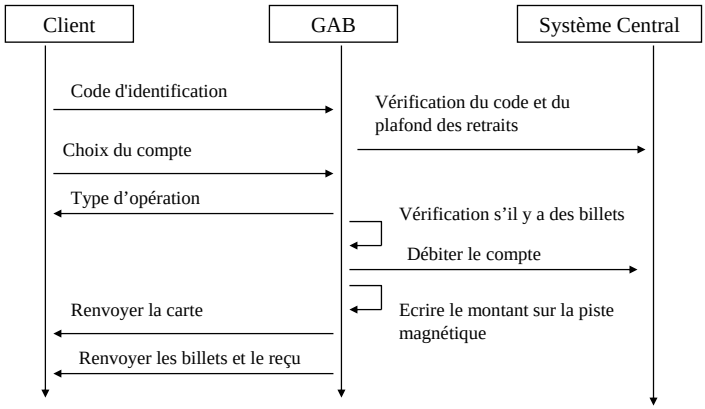
## Diagramme de cas d'utilisation

- Contexte global du projet

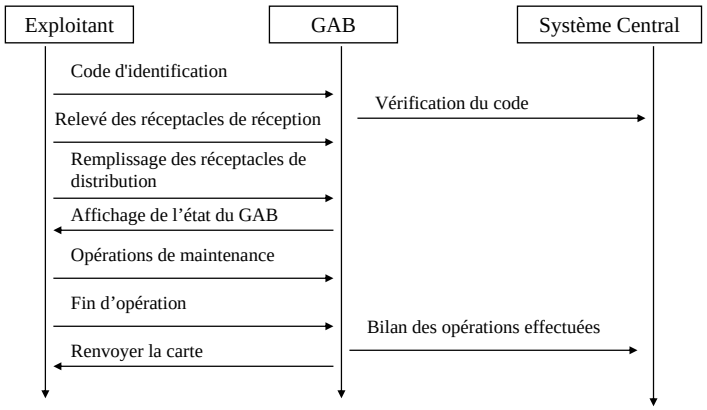





## Scénario de retrait de liquide par un client



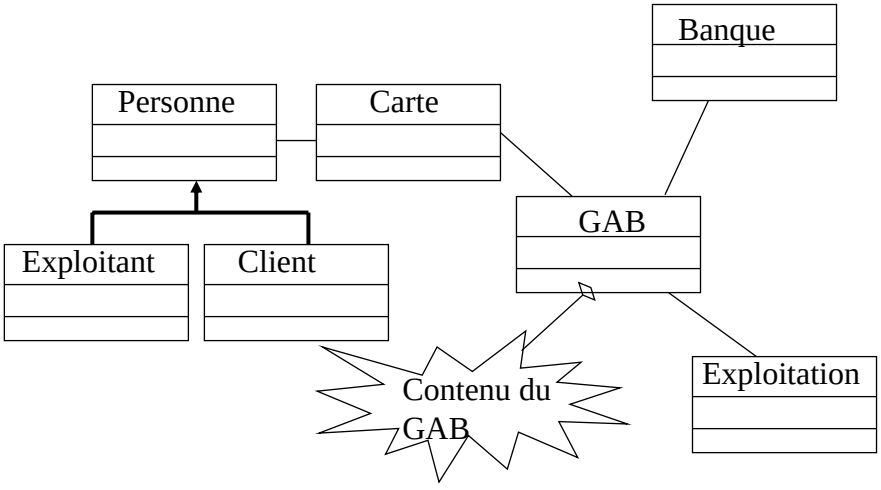
## Scénario d'exploitation du GAB





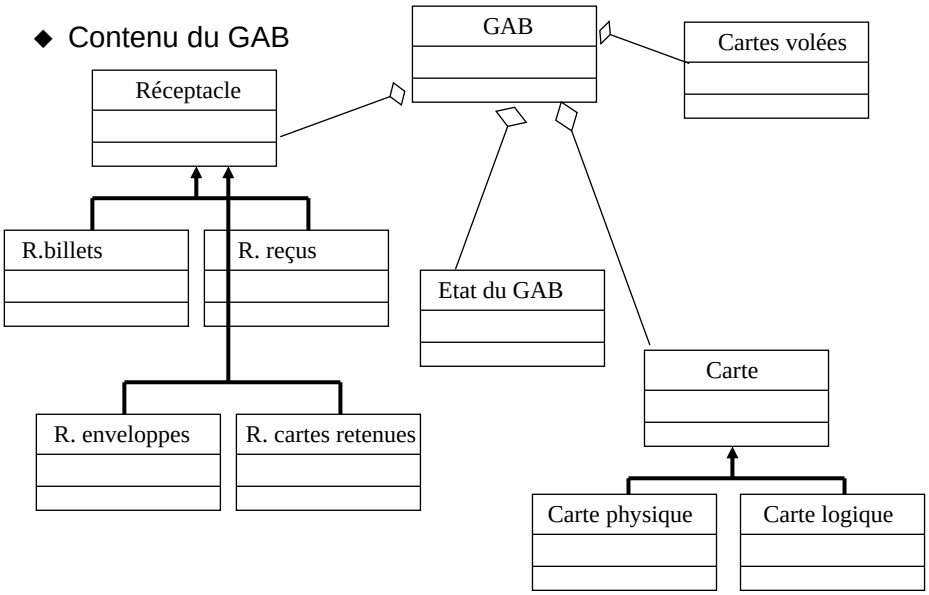

## Diagramme de classes

- Vue très partielle



## Diagramme de classes

- Contenu du GAB





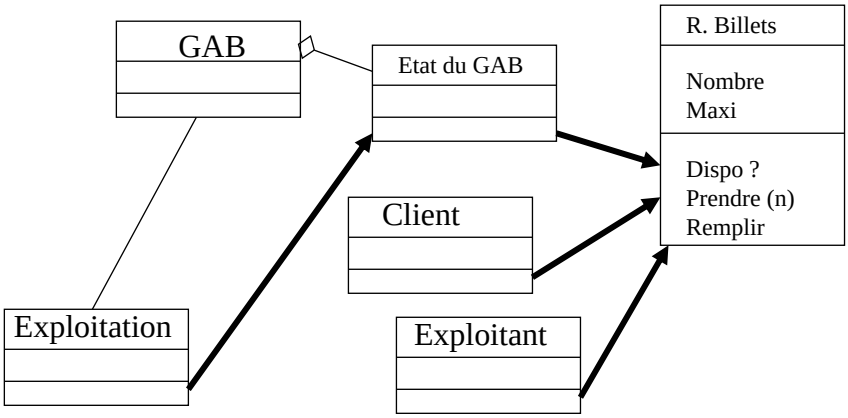

## Définition précise de chaque classe

- Réceptacle des Billets
  - **Attributs**
    - → **Nombre courant de billets**
    - → **Nombre maximum de billets**
  - **Services**
    - → **Nombre de billets disponibles**
    - → **Nombre de billets à prendre**
    - → **Nombre de billets (liasse) à mettre**
  - **Qui utilise les services :**
    - → **Nombre de billets disponibles - EXPLOITATION**
    - → **Nombre de billets à prendre - CLIENT**
    - → **Nombre de billets (liasse) à mettre - EXPLOITANT**

| R. Billets |
| --- |
| Nombre<br>Maxi |
| Dispo ?<br>Prendre (n)<br>Remplir |

## Diagramme de collaboration

- Description de liens entre classes
  - liens statiques (associations)
  - liens dynamiques (utilisation de services)






## UML en phase de conception

Objectif : passer du *quoi* au *comment*

Deux niveaux :

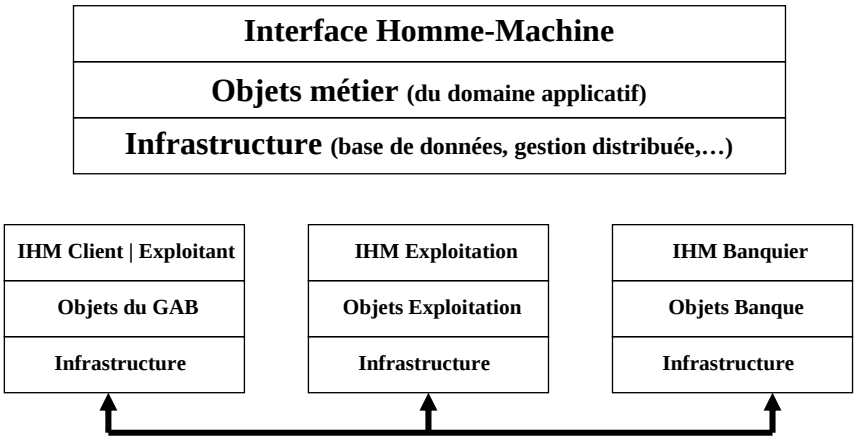
- Conception du système (conception d'architecture)
- Conception des objets (conception détaillée)

Deux approches :

- Architecture prédéfinie
- Architecture à concevoir

## La conception du système

- L'architecture prédéfinie d'un système est organisée en trois couches représentant trois sous-systèmes de " haut niveau " :

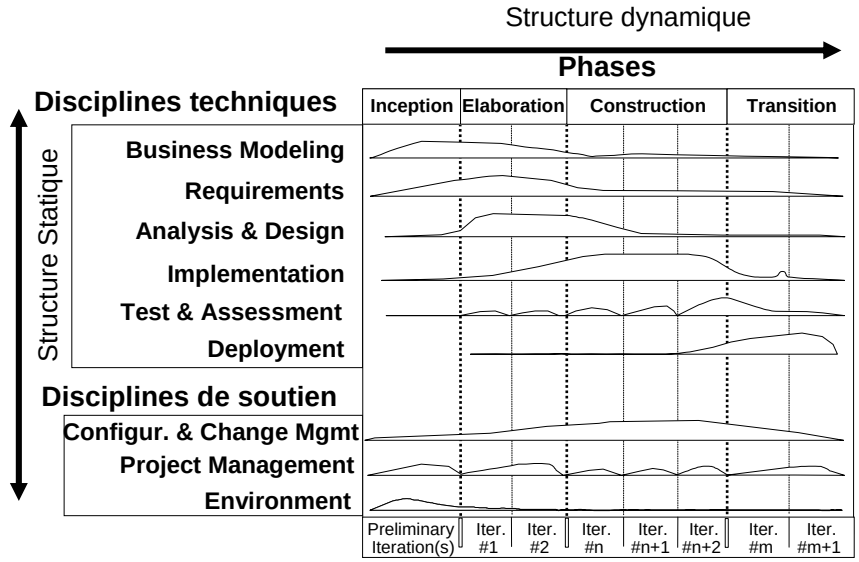





## La conception des objets (pour préparer l'implémentation)

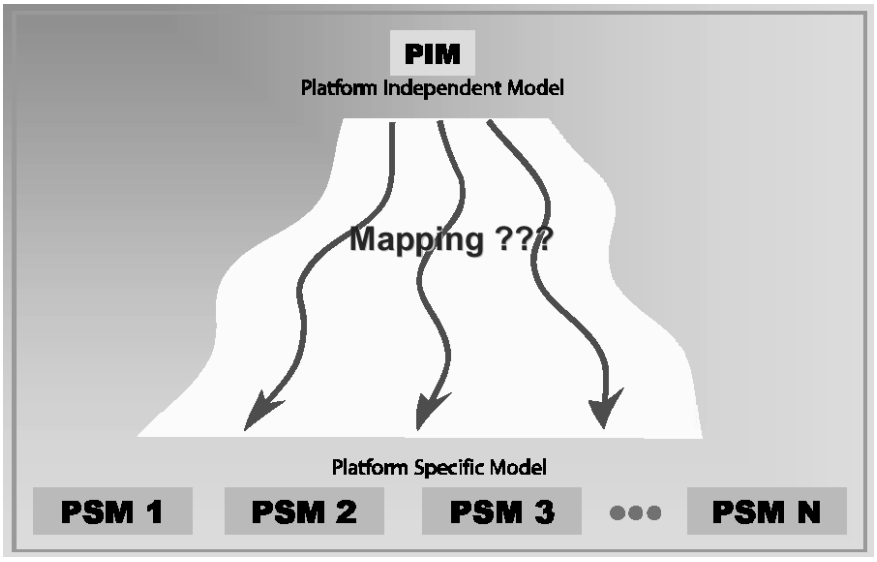
- Ajouter aux modèles objet de la spécification des détails liés à l'implémentation du système.
- Transposer le modèle objet de spécification par ajout ou par suppression d'entités en fonction du langage de programmation et du système de gestion de données utilisés.
- « Typer » des attributs : trouver les types primitifs (chaîne, entier,…) et des méthodes les manipulant

## RUP : deux dimensions




Centrale-Lyon - Cours POO (Info 2A) Penser Objet 23

## MDA : Model-Driven Architecture




Centrale-Lyon - Cours POO (Info 2A) Penser Objet 24

## Conclusion

- Approche objet n'est pas une mode
- C'est une première démarche permettant au logiciel de devenir un produit industriel
- Vous en saurez plus :
  - dans le cours d'approfondissement de 2° année « Développement de logiciels basé sur les processus »
  - dans l'option INFORMATIQUE de 3° année