MAI NFE103
Année 2013-2014

Méthodologies Orientées-Objet

F.-Y. Villemin (f-yv@cnam.fr)

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/MAI/index.html

Plan

- Les différentes méthodologies
  - Démarche
  - Cycle de vie
- Rational Unified Process (RUP)
- La méthode Layman
- Notre démarche
- Les méthodes agiles : Extreme programming (XP)

Méthodologie

Entrée :

une spécification floue, réduite, éventuellement inconsistante du projet

Sortie :

une description complète, consistante, compréhensible

- des caractéristiques
- du comportement
- du (des buts) du projet

Méthodologie

Une **méthode** se présente sous la forme de **phases**, associée à des concepts et notations adaptées à chaque phase

Les **phases** sont organisées dans un **cycle de vie**, comprenant plusieurs phase

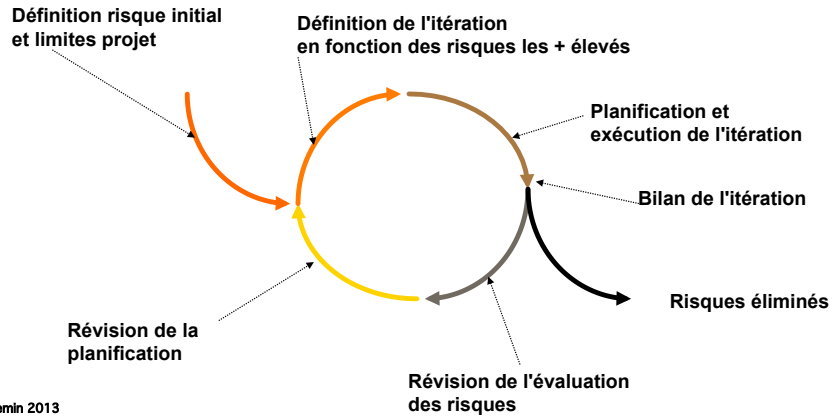
UML ne décrit pas de phases ou d'étapes

En fonction du type de projet, et du type de risque, il convient de définir des étapes appropriées

Les **projets Objets** se caractérisent par un **cycle** de développement **incrémental et itératif**

RUP : Cycle de vie

Cycle de vie guidé par les risques est proposé par la société Rational (Groupe IBM) sous le nom de "Rational Unified Process" (RUP) :



RUP : Phases

Les phases du cycle de développement:

- Initialisation : Définition du problème
- Analyse : Planification des activités, Affectation des ressources, Cahier des Charges, Analyse
- Développement : Développer le logiciel au travers d'une série d'itérations incrémentales (Conception Générale, Conception détaillée, Implantation, Tests)
- Recettage et déploiement

RUP : Phase Initialisation

- Définition de l'étendue du projet
- Poser le problème : Quel est le système dont l'utilisateur a besoin ?
- Identifier les acteurs : utilisateurs, gestionnaires du système, plates-formes, interfaces avec autres applications
- Identifier les cas d'utilisations

  Préciser les cas d'utilisation les plus importants, en fonction du risque, sous forme de scénarios (itération sur les scénarios)

RUP : Phase Analyse

- Analyse du domaine :

  Modèle Objet statique

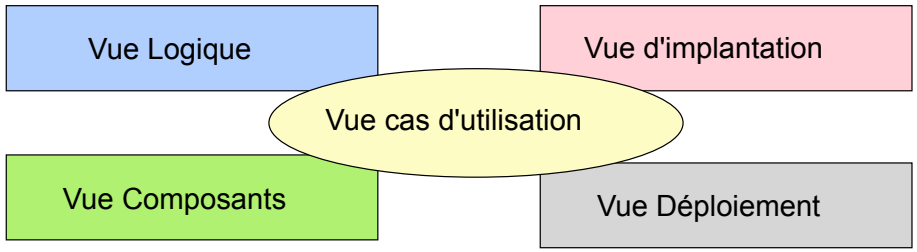
  Scénarios (Diagramme de séquences)
- Définition de l'Architecture

  L'architecture est la façon d'organiser les objets pour qu'ils réalisent les fonctionnalités de l'application par leurs collaborations
- Planification du Développement

RUP : Définition de l'Architecture

La qualité d'un logiciel est celle de son architecture

Architecture en couches et "vues" de Ph. Krutchen :



RUP : Définition de l'Architecture

- **Vue Cas d'utilisation :**
  Décrit le système comme un ensemble de transactions du point de vue de l'utilisateur. Créée lors de la phase d'initialisation
- **Vue Logique :**
  Contient une collection de conteneur, classes et associations
  Créée lors de la phase d'analyse et raffinée lors de la phase de développement
- **Vue Composants :**
  Contient une collection de conteneur, et de programmes

RUP : Définition de l'Architecture

- **Vue Déploiement :**
  Décrit l'architecture matérielle. Contient une collection d'unités et de processus
  Créée lors de la phase d'analyse
- **Vue Implantation :**
  Contient une collection de modules et sous-modules (conteneur)
  Créée lors de la phase d'analyse et affinée lors de la phase de développement

RUP : Planification

1. Identifier les risques principaux
2. Sélectionner un petit nombre de scénarios couvrant ces risques. Les scénarios sont classés par ordre de priorité en fonction des risques, de leur importance pour le client et du point de vue fonctionnel
3. Sélectionner les classes et leur relations à implanter en analysant les scénarios
4. implanter les classes et relations sélectionnées

RUP : Planification

5. Définir le plan de test
6. Tests : vérifier que les fonctionnalités des scénarios sont correctement réalisées
7. Tests d'intégration avec le résultat des itérations précédentes
8. Bilan de l'itération et définition des révisions à apporter lors de l'itération suivante

RUP : Phase Développement

1. Identifier les classes et relations à implanter
2. Compléter les classes et relations sélectionnées:
   - Typage des attributs
   - Méthodes et leurs signatures
   - Ajout de classes d'implantation (conteneurs, contrôleurs)
   - Choix de conception pour les relations (navigation, ...)
   - Choix de conception concernant l'héritage (classes abstraites, délégation, ..)

RUP : Phase Développement

3. Codage
4. Création / Mise à jour de la documentation
5. Tests des créés/modifiés par l'itération
6. Tests d'intégration des éléments créés/modifiés par l'itération

Exemple d'utilisation de la méthodologie RUP

Inscription aux cours d'une université

Existant :

- Les étudiants remplissent des imprimés et les envoient au service d'inscription. Des employés saisissent les sélections des étudiants dans une base de données, et fournissent les emplois du temps pour les étudiants.
- L'université décide d'utiliser un système d'inscription en temps réel. Ce système doit être utilisé par les professeurs pour leur indiquer les cours qu'ils doivent assurer, par les étudiants pour choisir leurs cours, et par le service d'inscription pour compléter le processus d'inscription.

Exemple d'utilisation de la méthodologie RUP

Définition du problème :

- Au début de chaque semestre les étudiants doivent obtenir un catalogue des cours proposés : contenu du cours, professeur, département, et pré-requis
- Le nouveau système doit permettre aux étudiants de sélectionner leurs cours pour le semestre. En plus, chaque étudiant doit indiquer un choix alternatif pour chaque cours choisit, au cas où ce cours serait supprimé. Aucun cours ne doit avoir plus de vingt étudiants. Un cours auquel moins de dix étudiants sont inscrit doit être supprimé

Exemple d'utilisation de la méthodologie RUP

- Une fois que le processus d'inscription par l'étudiant est terminé, le système fournit les droits d'inscription à percevoir pour le semestre.
- Les professeurs doivent pouvoir renseigner le système sur les cours qu'ils assurent. Ils doivent aussi savoir quels étudiants se sont enregistrés à leurs cours.
- Lors de chaque début de semestre, il est défini une période de temps où les étudiants peuvent modifier leurs choix. Les étudiants doivent pouvoir accéder au système pour modifier leurs choix. Pendant cette période de temps, le système d'inscription doit créditer les étudiants pour les cours qu'ils ont annulés

Phase d'Initialisation

Définition des risques :

- Le risque principal est constitué par l'interface IHM (bonne acceptation par les étudiants et professeurs)

Identifier les acteurs :

- Etudiant
- Professeur
- Employé du service enregistrement
- Système de facturation des droits d'inscription (externe)

Cas d'utilisation

Etudiant

- Inscription aux cours

Professeur

- Sélection des cours à assurer
- Lister les étudiants inscrits à un cours

Employé du service inscription

- Génération du catalogue des cours
- Mise à jour des informations professeurs
- Mise à jour des informations étudiants
- Mise à jour des listes de cours

Diagramme des cas d'utilisation

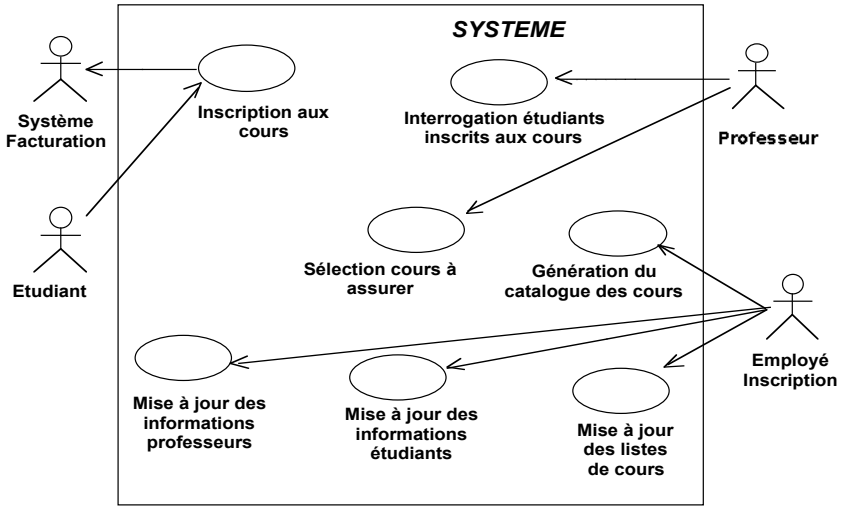


Diagramme des cas d'utilisation

Un cas d'utilisation peut contenir des diagrammes de cas d'utilisation, formant ainsi une arborescence

En pratique, l'arborescence des cas d'utilisations correspondra à l'arborescence du menu de l'application

Les fonctionnalités élémentaires classiques (ajouter un item, modifier un item, supprimer un item) correspondront aux scénarii contenus dans un cas d'utilisation

Diagramme des cas d'utilisation

Le cas d'utilisation "Inscription aux cours" contient :

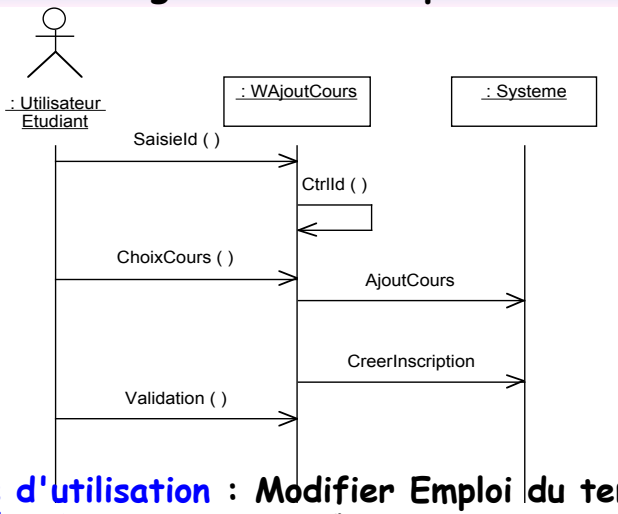
- Le cas d'utilisation "Création emploi du temps"
- Le cas d'utilisation "Modifier emploi du temps" contient :
  - Le cas d'utilisation "Supprimer un cours"
  - Le cas d'utilisation "Ajouter un cours"

Scénarii

On décide de développer les scénarii correspondants aux cas d'utilisation :

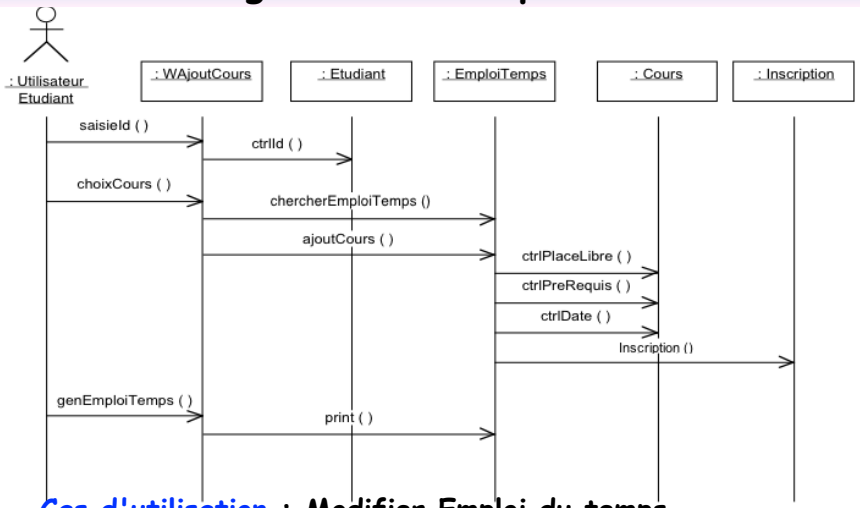
- Le cas d'utilisation "Inscription aux cours" :
  - Scénario : Créer un emploi du temps
  - Scénario : Lister un emploi du temps
- Le cas d'utilisation "Modifier un emploi du temps" :
  - Scénario : Supprimer un cours
  - Scénario : Ajouter un cours

Diagramme de séquences



Cas d'utilisation : Modifier Emploi du temps
Scénario : Ajouter un Cours

Diagramme de séquences

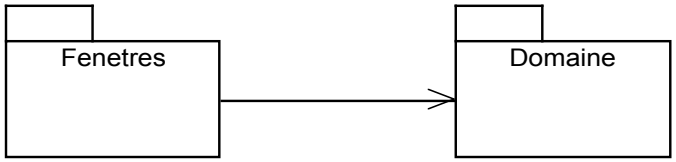


Cas d'utilisation : Modifier Emploi du temps
Scénario : Ajouter un Cours

Modèle Objet

On définit deux paquetages :

- Domaine
- Fenêtres

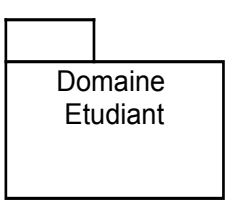


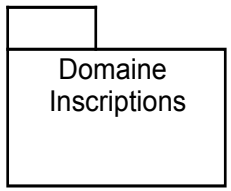
Le paquetage Fenêtre utilise le paquetage Domaine

Modèle Objet

Le paquetage Domaine contient :

- Domaine Etudiant
- Domaine Inscription

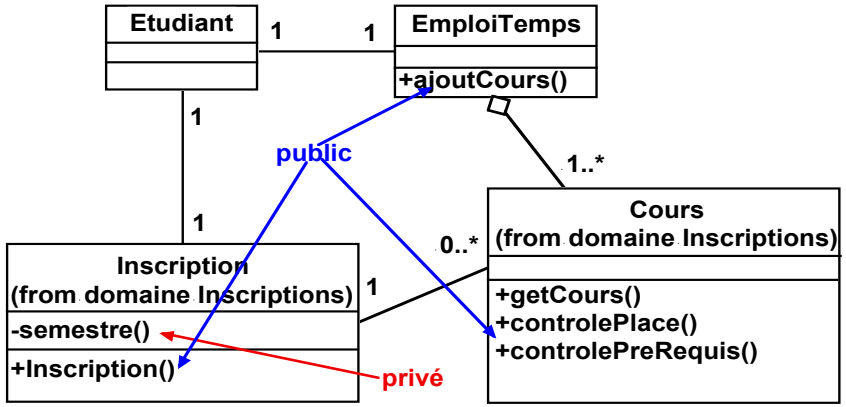




Modèle Objet

Le paquetage Domaine Etudiant contient les classes:

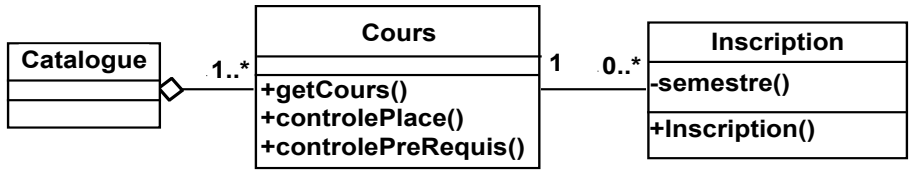
- Etudiant
- EmploiTemps



Modèle Objet

Le conteneur Domaine Inscription contient les classes :

- Catalogue
- Cours
- Inscription



Modèle Objet

Le conteneur Fenêtres contient la classe :

WModifCours

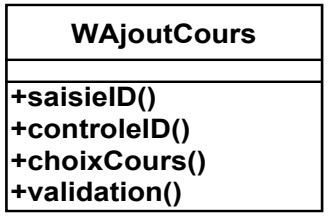
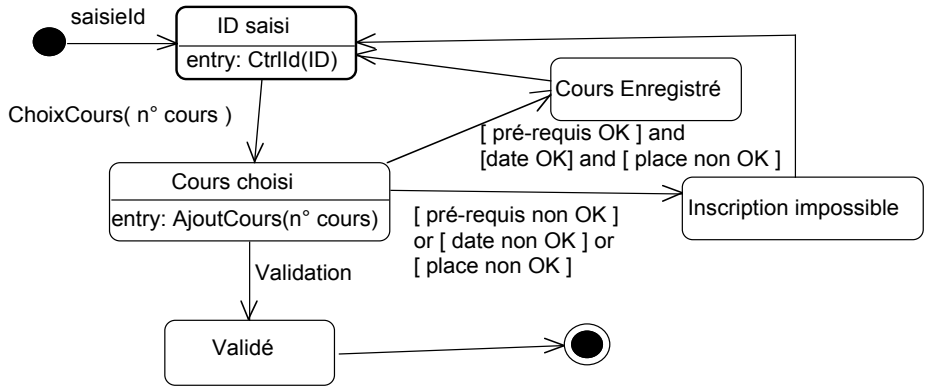


Diagramme d'Etats

Diagramme d'états de la classe Cours :



Démarche RUP

Dans la démarche RUP :

- Les objets (et donc leur classes), comme les méthodes, sont découverts en documentant les scénarios
  - C'est une démarche guidée par les scénarios
  - Remarque : le placement des méthodes sur les classes n'est pas toujours évident
  - Cette démarche est préférable lorsque l'étude des traitements doit guider l'analyse

Dans les autres cas, la production du modèle de classes doit être faite avant l'étude des scenarii, comme le propose C. Larman et notre démarche

La méthode Larman

- Diagrammes de cas d'utilisations
- Diagramme de classes
- Diagramme d'interactions
  - Diagrammes de séquences
  - Contrats
  - Diagrammes de collaboration
- Diagramme d'états
- Diagrammes d'activités
- Diagramme de composants
- Diagramme de déploiement

La méthode Larman : Cas d'utilisation

Cas d'utilisation : nom du cas d'utilisation
Acteurs : intervenants dans le cas d'utilisation (utilisations, évènements externes, déclencheurs internes)
But : but du cas d'utilisation
Résumé: résumé de haut niveau du cas d'utilisation
Type : primaire, secondaire, ou optionnel
Références croisées : cas d'utilisation et fonctions systèmes reliées
Scénario typique
Scénarii alternatifs

La méthode Larman : classes

Au niveau analyse :

- Une classe "représente une entité (conceptuelle) dans le domaine d'affaires de l'application (exemple : Client) au niveau analyse"
- pas d'opérations associées aux classes, la distribution de la fonctionnalité est une décision de conception et des heuristiques peuvent être utilisées (CRC: Collaboration – Responsibility)

La méthode Larman : séquences

Au niveau analyse :

- On représente le système par une boite noire
- On ne détaille pas les échanges entres objets à l'intérieur du système
- L'échange entre objets implique une distribution de responsabilités, qui est une décision de conception
- Les diagrammes de séquence ne sont rien d'autre que des transcriptions (plus précises) des scénarios des cas d'utilisation, avec plus d'information (nom des opérations, paramètres)

La méthode Larman : contrats

Les contrats décrivent le comportement du système et détaillées en termes de changements d'état des objets dans le modèle de domaine, après une opération du système a exécuté

Les contrats sont des descriptions fonctionnelles (et non algorithmiques) des opérations principales du système :

- Une opération système est une opération "globale" du système en réponse à des interactions de l'utilisation
- A chaque évènement reçu par le système, correspond une opération (la réponse du système à cet évènement)

La méthode Larman : contrats

Nom: du contrat
Responsabilités: une description informelle de ce que l'opération doit faire / accomplir
Type (concept, interface, implantation)
Références croisées: avec des fonctions systèmes, des cas d'utilisation, des diagrammes de séquences
Notes: informations sur l'algorithme, conception : cas exceptionnels
Sortie: autre que les sorties niveau interface (e.g. messages, menu, etc.)
Pré-conditions: hypothèses sur l'état du système avant l'appel
Post-conditions: état du système après l'exécution de l'opération

La méthode Larman : contrats

Comment construire un contrat :

1. Identifier les opérations systèmes à partir des diagrammes de séquence
2. Pour chaque opération, construire un contrat
3. Commencer par écrire la section responsabilités
4. Décrire les post-conditions en termes de:
   - Création ou destruction d'objet
   - Modification d'attributs
   - Création ou destruction d'associations

Notre démarche

Dans le cas d'une application de gestion, il est préférable de faire une analyse guidée par les données

- Spécifier le modèle Objet statique avant de définir les scénarios
- Pour ne pas se poser le problème du placement des méthodes avant que l'architecture soit définie, on définit une classe système qui reçoit des événements de la fenêtre

Notre démarche

Notre démarche s'inspire de celle de C. Larman,dans le cycle de développement :

- Diagramme de cas d'utilisations : pour définir de l'étendue du projet et identifier les acteurs
- Diagramme de classes : par une analyse textuelle
- Pas de diagramme d'interactions, juste des diagrammes de séquences et des diagrammes de collaboration détaillés rendant inutiles les contrats : les scenarii sont rédigés précisément, puis formalisés en diagrammes de séquences (collaboration)

Notre démarche

Notre méthode s'est montrée dans la pratique plus aisée à mettre en œuvre principalement pour des analystes de culture MERISE

1. Analyse du domaine par analyse textuelle :
   - Recherche des candidats classes
   - Revue des candidats classes
   - Recherche des candidats attributs pour chaque classe retenue
   - Revue des candidats attributs
   - Recherche des candidats associations entre classes retenues
   - Revue des candidats associations

Notre démarche

2. Traitement des objets complexes
3. Recherche des agrégations
4. Recherche des généralisations
5. Recherche des spécialisations
6. Vérification de l'utilisation de l'héritage

Analyse du domaine par analyse textuelle

(Abbot R. J., "Program Design by Informal English Descriptions", Com. of the ACM, vol. 26 n°11, 1983, PP. 882-894)

- Nom propre ⇒ instance d'une classe
  "Paul est élève au CNAM" ⇒ Paul est instance de la classe Elève
- Nom commun {objets possédant des caractéristiques communes} ⇒ classe
  " Les Unités de Valeurs du CNAM …" ⇒ classe Unité_de_Valeur
- Nom commun (suivis d'une forme possessive) ⇒ attribut
  "La couleur de la voiture" ⇒ couleur est attribut de la classe Voiture

Analyse du domaine par analyse textuelle

- Verbe (association entre objets ou opération effectuée sur des objets) ⇒ association ou méthode
  "Le Président ouvre la session" ⇒ ouvrir méthode de la classe Session (Le responsable est Président, sujet du verbe ouvre)
  "Le client prend un abonnement à une revue" ⇒ prend indique une association "abonnement" entre classes Client et Revue
- Adjectif ⇒ valeur d'un attribut de l'objet qualifié
  "Les véhicules rouges" ⇒ rouge est la valeur d'un attribut (couleur) de la classe Véhicule

Analyse du domaine par analyse textuelle

Candidat Classe ⇒
nom commun précédé d'un article indéfini (ou équivalent : quelque, certain, un membre de, un ensemble de…)

Candidat Attribut (d'une classe retenue) ⇒
1. nom commun lié au nom de la classe dans une forme possessive
2. nom commun correspondant à la nature d'un adjectif lié au nom de la classe

Candidat Association (entre deux classes retenues) ⇒
1. verbe (autre qu' être et avoir) entrant dans une phrase liant les deux noms de classe
   "Le client achète un article"
2. nom commun correspondant à verbe substantivé liant deux noms de classe
   "L'achat de l'article par le client"

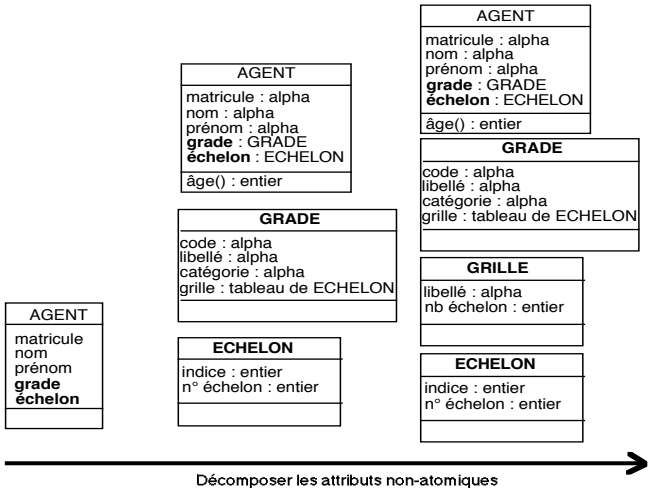
Analyse du domaine par analyse textuelle

Candidat Association (entre 2 classes retenues) ⇒

1. verbe (autre qu'être et avoir) entrant dans une phrase liant les deux noms de classe
   "Le client achète un article"
2. nom commun correspondant à verbe substantivé lié au deux noms de classe
   "L'achat de l'article par le client"

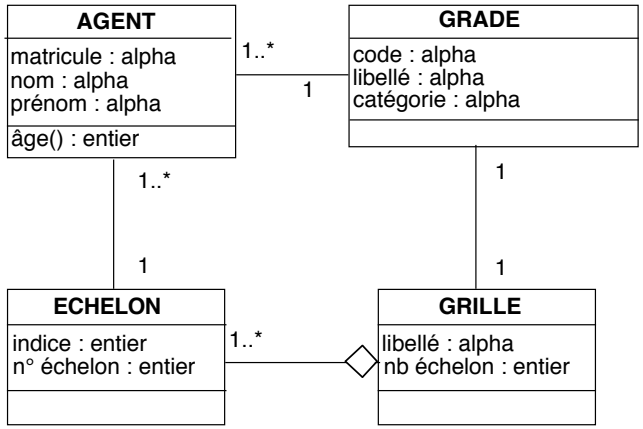
Objet Complexe

Décomposition des attributs non-atomiques d'un objet



Objet Complexe

Définition d'un objet complexe par composition d'objets plus simples :



Reconnaître une agrégation

Une association est une agrégation si :

1. On peut utiliser l'expression "partie-de" ? Des opérations appliquées sur le composé sont-elles appliquées automatiquement aux composants (le composé gère ses composants) ?
   Exemple : totaliser une facture
2. Des valeurs d'attributs sont-elles propagées du composé vers des composants.
   Exemple : la date d'une facture "concerne" les lignes factures
3. Il y a une asymétrie intrinsèque dans l'association dans laquelle une classe est subordonnée à l'autre ?

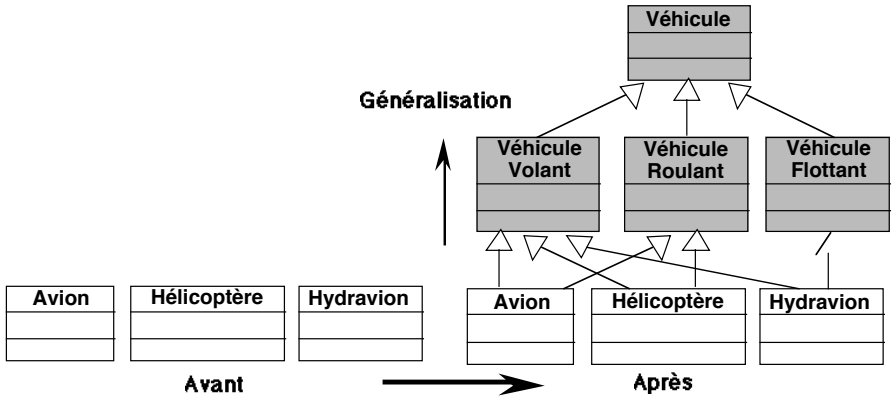
Héritage

Généralisation et héritage permettent de partager les points communs entre les classes

Généralisation : abstraction des caractéristiques communes à plusieurs classes d'objets

Le terme généralisation fait référence à la relation qui lie les sous-classes à leur super-classes

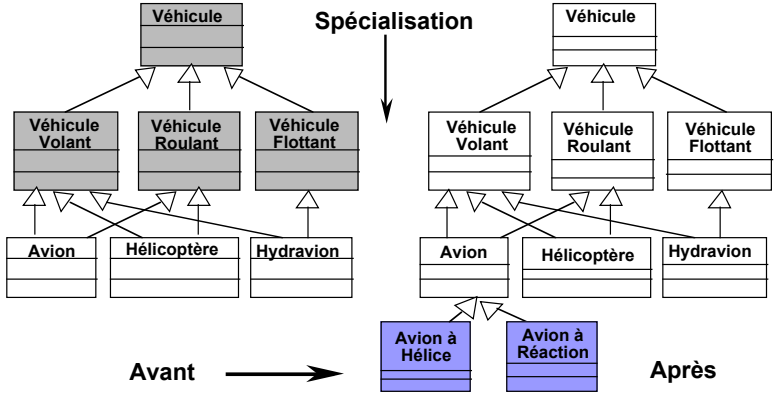
Héritage



Héritage

Spécialisation

Définir par dérivation de nouvelles abstractions contenant des caractéristiques supplémentaires



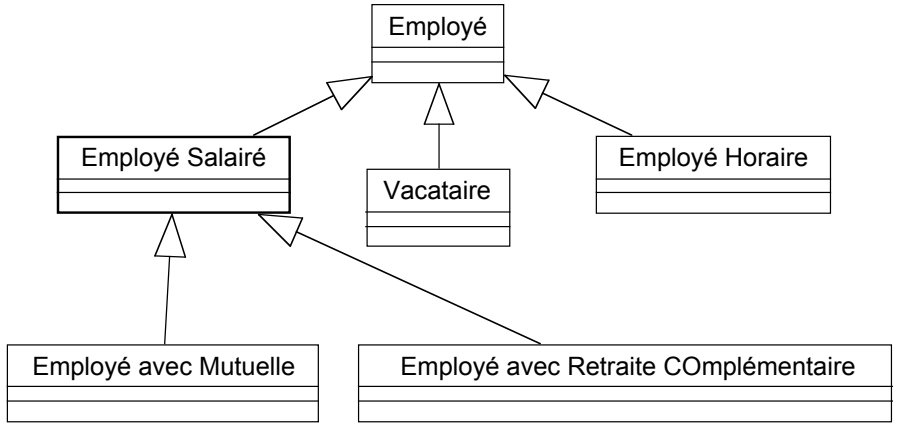
Règles d'utilisation de l'héritage

Il y a un problème d'utilisation de l'héritage lorsque il y a un nombre important de sous-classes possibles (≥5), ou la définition des spécialisations n'est pas un invariant du domaine (elle sera sujette à des remises en cause)

Une technique à utiliser :

- Délégation
- Croisement des "points de vue" de spécialisation

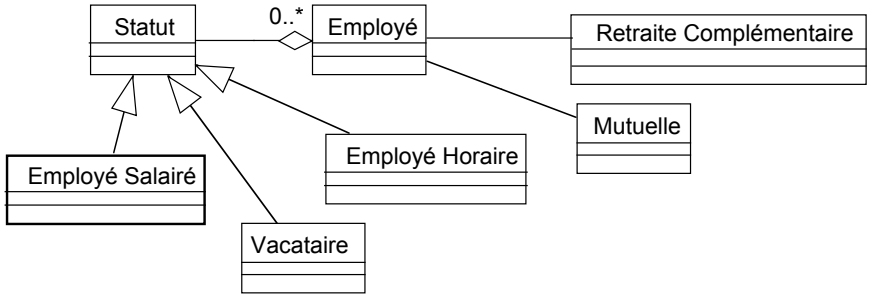
Règles d'utilisation de l'héritage



Règles d'utilisation de l'héritage

Délégation utilisant l'agrégation de rôles :

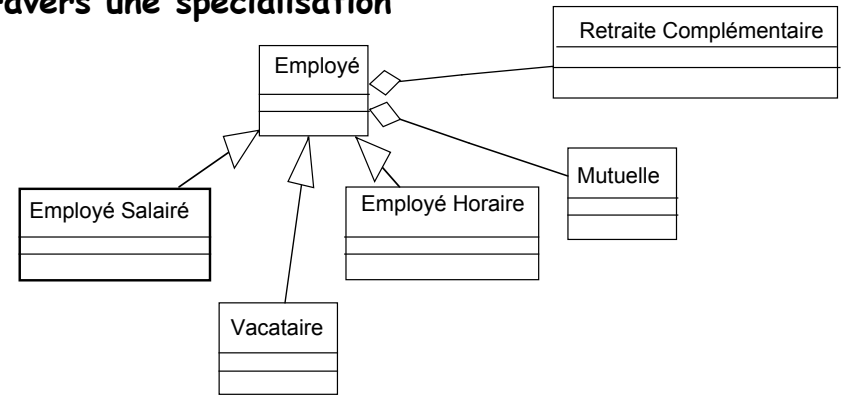
Une super classe avec des spécialisations indépendantes peut être remaniée comme un agrégat pour lequel chaque composant remplace une spécialisation



Règles d'utilisation de l'héritage

Délégation : hériter de la classe la plus importante et déléguer le reste

Cette approche préserve l'identité de l'héritage à travers une spécialisation



Les méthodes agiles

Une méthode agile est une méthode de développement informatique permettant de concevoir des logiciels en impliquant au maximum le demandeur (client) :

- plus grande réactivité à ses demandes
- plus pragmatiques que les méthodes traditionnelles
- recherche de la satisfaction réelle du besoin du client

La notion de méthode agile est née d'un manifeste signé par 17 personnalités (parmi lesquelles Ward Cunningham, l'inventeur du Wiki), créateurs de méthodes ou dirigeants de sociétés

(d'après "Les méthodes agiles" : http://fr.wikipedia.org/)

Les méthodes "agiles"

Le manifeste prône 4 "valeurs fondamentales" :

1) L'équipe ("Personnes et interaction plutôt que processus et outils") : Avoir une équipe soudée et qui communique : la communication est une notion fondamentale
2) L'application ("Logiciel fonctionnel plutôt que documentation complète") : L'application doit fonctionner, la documentation technique est secondaire (il est préférable de commenter abondamment le code lui-même)
3) La collaboration ("Collaboration avec le client plutôt que négociation de contrat") : Le client doit être impliqué dans le développement, il fournir un feed-back continu sur l'adaptation du logiciel à ses attentes
4) L'acceptation du changement ("Réagir au changement plutôt que suivre un plan") : La planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. Les premières *releases* du logiciel vont souvent provoquer des demandes d'évolution

Principes des méthodes agiles

Ces 4 valeurs conduisent à 12 principes généraux communs à toutes les méthodes agiles :

1) "Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles"
2) "Le changement est bienvenu, même tardivement dans le developpement. Les processus agiles exploitent le changement comme avantage compétitif pour le client"
3) "Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte"
4) "Les gens de l'art et les développeurs doivent collaborer quotidiennement au projet".
5) "Batissez le projet autour de personnes motivées. Donnez leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail"
6) "La méthode la plus efficace de transmettre l'information est une conversation en face à face"

Principes des méthodes agiles

7) "Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet"
8) "Les processus agiles promeuvent un rythme de développement soutenable. Sponsors, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment"
9) "Une attention continue à l'excellence technique et à a qualité de la conception améliore l'agilité"
10) "La simplicité - l'art de maximiser la quantité de travail qu'il est inutile de faire - est essentielle"
11) "Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent"
12) "À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens"

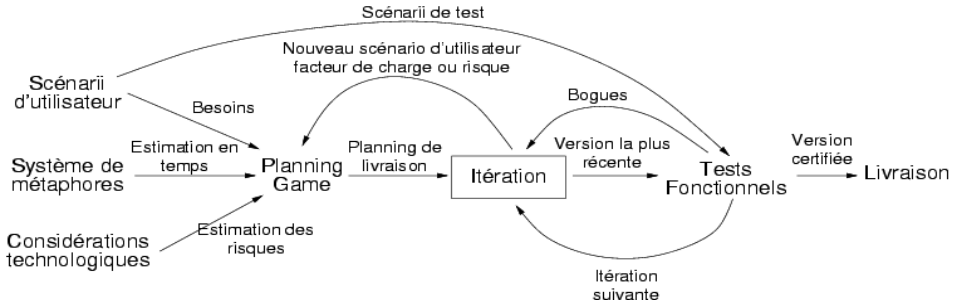
Les méthodes agiles

Méthodes agiles :

- Extreme programming (XP)
- Dynamic software development method (DSDM)
- Adaptive software development (ASD)
- Scrum
- Feature driven development
- Crystal clear
- ...

Extreme programming (XP)

Le cycle de l'Extreme Programming :



La méthode XP

Le jeu de la planification

Release courtes :
- Planification sur un mois ou deux préférable à 6 mois ou un an

Métaphore :
- Chaque projet est guidé par une métaphore

Conception simple

Tests

Refactorisation :
- Simplification de la conception après ajout d'une fonctionnalité
- La nouvelle conception doit passer les tests

La méthode XP

Programmation par paire

Appartenance collective
- Chacun peut modifier tout morceau de code à chaque instant

Intégration continue
- Le code est intégré et testé après quelques heures, une journée au plus

40 heures par semaines

Un client sur site

Conventions de codage
- adoptées par toute l'équipe

La méthode XP

Les gestionnaires décident :
- La portée du logiciel
- Priorité des tâches
- Composition des releases
- Jalons

Les programmeurs décident :
- Estimations de temps
- Conséquences des choix techniques (Ex: BD)
- Organisation de l'équipe
- Calendrier détaillé (tâches à haut risque d'abord, flexible pour prendre en compte les besoins du client)

La méthode XP

Programmation par paire :
- 1 ordinateur : 2 programmeurs
- L'un pense au code nécessaire pour une méthode
- L'autre pense globalement :
  - l'approche a t'elle marché
  - Quels sont les autres cas de tests qui ne marchent toujours pas.
  - N'y a t'il pas une solution plus globale pour résoudre tous ces problèmes ?

Bibliographie

"Le guide de l'utilisateur UML", Grady Booch, James Rumbaugh et Yvar Jacobson, Eyrolles 2000

"Applying UML and Patterns", Craig Larman, Addison Wesley/Pearson Education, 1999

"Extreme Programming Explained: Embrace Change", Kent Beck, Addison Wesley/Pearson Education, 2000

"Agile and Iterative Development: A Manager's Guide", Craig Larman, Addison Wesley/Pearson Education, 2003