

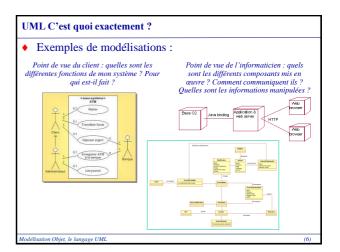
UML C'est quoi exactement ?

- ◆ Comme disait les connaisseurs ... un bon dessin vaut mieux qu'un long discours!
- Quel sont les premiers outils que l'on utilise pour construire une maison, un immeuble, une ville :
 - La pelle et le seau ? Le crayon et le papier ?
- Quand un système est complexe, il est nécessaire de pouvoir en faire des représentations simples, axées sur un point de vue particulier.
- Exemples : le plan électrique d'une habitation, une carte routière, ...

Modélisation Objet, le langage UML

UML C'est quoi exactement ?

- En informatique aussi, les systèmes peuvent être complexes!
- Il devient alors nécessaire, quelque soit la démarche choisie, de faire des représentations – pour un point de vue particulier – du système que l'on construit.
- On parle alors de modèles et de modélisation.
- Les diagrammes UML permettent de mieux appréhender un système ou un logiciel complexe grâce à une représentation visuelle de l'architecture du code et des relations entre les différents composants.



UML C'est quoi exactement ?

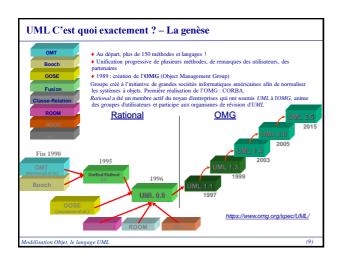
- Pour partager un modèle avec d'autres personnes, il faut s'assurer au préalable :
 - que les personnes comprennent les conventions de représentation et les notations utilisées,
 - qu'il ne pourra pas y avoir d'ambiguïté sur l'interprétation du
- D'où l'émergence de langages de modélisation de systèmes informatiques.
- Un langage de modélisation doit définir sans ambiguïté les concepts utilisables, leur représentation, les règles et contraintes associées.

UML C'est quoi exactement ?

UML = Unified Modeling Language Langage unifié pour la modélisation

Langage de modélisation objet, indépendant de la méthode utilisée. UML est un langage pour :

- · Comprendre et décrire un problème,
- Spécifier un système, de manière précise et complète, sans
- Concevoir et construire des solutions, une partie du code des diagrammes de classes peut être généré automatiquement,
- Documenter un système, les différents diagrammes, notes, contraintes, exigences sont conservés dans un documen
- Communiquer.
- MODELING Unifié: mise en commun et convergence de bonnes p de langages antérieurs, émergence d'un standard.
- Pourquoi modéliser ? Pourquoi ne pas développer tout de suite



UML c'est quoi exactement ? - La genèse

- UML a été « boosté » par l'essor de la programmation orientée objet.
- Historique de l'approche orientée objet :
 - Simula (1967),
 - Smalltalk (1976),
 - C++ (1985),
 - Java (1995),
 - .net ...
- UML permet de modéliser une application selon une vision objet, indépendamment du langage de programmation.

Modélisation Objet, le langage UML

(10)

Order treprésenter une classe ? Un objet ? VI. Vehicule Transportation = AA. 115.AA determinationation = 10.017.017 proprieture = Jacques Durand Tout est défini dans la spécification UML! • http://www.omg.org/spec/UML/ Modélibation Objet, le language UML. (11)

UML c'est quoi exactement ? - La portée

- UML reste au niveau d'un langage et ne propose pas de processus de développement
 - ni ordonnancement des tâches,
 - ni répartition des responsabilités,
 - ni règles de mise en œuvre.
- Il existe de (très) nombreux outils pour faire de la modélisation UML: StarUML, ArgoUML, Papyrus,...
- Certains ouvrages et AGL basés sur UML proposent un processus en plus de UML.
 - Exemple : le processus unifié UP.

Modélisation Objet, le langage UML

(12)

UML c'est quoi exactement ? - Le méta-modèle

- UML est bien plus qu'un outil pour dessiner des représentations mentales!
- La notation graphique n'est que le support du langage
- UML repose sur un <u>méta-modèle</u>, qui normalise la sémantique de l'ensemble des concepts.
- UML utile pour mettre les nouveaux développeurs à niveau sur les projets et pour concevoir de nouveaux projets à partir de zéro. La nature visuelle de ces modèles les rend beaucoup plus faciles à comprendre que l'alternative consistant à devoir lire et appréhender potentiellement des centaines de milliers de lignes de code pour comprendre un nouveau système.
- UML Aide à concevoir des projets d'ingénierie complexes

Modelisation Objet, le langage UML

(13)

UML c'est quoi exactement ? - En résumé

- UML est un langage de modélisation objet
- UML n'est pas une méthode
- UML convient pour tous les types de systèmes, tous les domaines métiers, et tous les processus de développement
- UML est dans le domaine public
- Les concepts véhiculés dans UML sont définis et non équivoques

Modélisation Objet, le langage UML

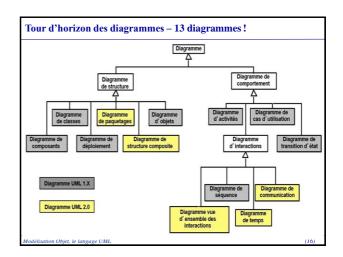
(14)

Tour d'horizon des diagrammes - 13 diagrammes!

- Historiquement UML proposait 9 types de diagrammes (UML 1.x).
- UML 2 a enrichi les concepts des diagrammes existants et a ajouté 4 nouveaux types de diagrammes.
- Les diagrammes manipulent des concepts parfois communs (ex. classe, objet, ...), parfois spécifiques (ex. cas d'utilisation).
- Quelle différence entre un modèle et un diagramme ?

Modélisation Obiet, le langage UML

(15)



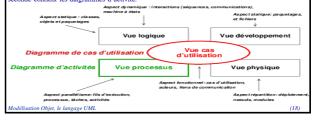
Tour d'horizon des diagrammes - 13 diagrammes!

- UML est une grande boîte à outils, et comme toute boîte à outils
 - On utilise pas tout ...
 - · Certains outils servent plus souvent que d'autres
 - · Dans chaque situation il faut choisir le bon outil!
- En tant que langage, il faut aussi s'assurer que l'on va être compris
- Modèle pour appréhender la réalisation d'un système informatique
- Diagrammes UML qui permettent d'aider à la réflexion, à la discussion (clients, architectes, développeurs, etc.)
- Pas de solution unique mais un ensemble de solutions plus ou moins acceptables suivant les contraintes du client, et les logiciels et matériels disponibles
- Une solution acceptable est obtenue par itérations successives

Modèle de l'analyse Vision de l'extérieur du système, le problème :

- Vue cas d'utilisation = ce que fait le système, les fonctionnalités
- Vue processus = ce que fait le système, les règles de gestion

L'élément pivot du modèle du système informatique est la vue cas d'utilisation. Cette vue est la plus proche du client et des utilisateurs finaux. C'est elle que ces derniers comprennent le mieux. Elle est complétée par la vue processus qui décrit comment le système répond aux stimulus externes. Cette vue processus présente les processus métier, par exemple les règles de gestion d'un système d'information. La première vue est composée des diagrammes de cas d'utilisation alors que la seconde contient les diagrammes d'activité.



Diagrammes

Nous allons nous concentrer sur les 5 diagrammes suivants : Cas d'utilisation, Classes, Séquence, États, Activités.

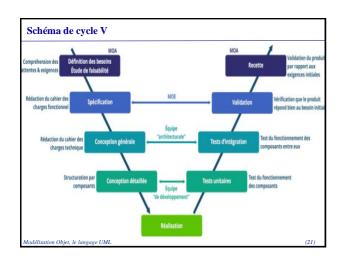
UML2 apporte 4 nouveaux diagrammes:

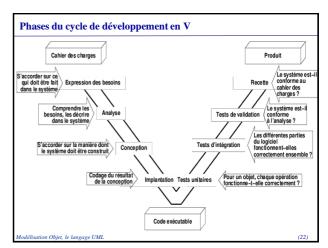
- diagramme de structure composite (composite structure diagram) : permet de décrire la structure interne d'un objet complexe lors de son exécution (au run-time - décrire l'exécution du programme), dont ses points d'interaction avec le reste du système
- diagramme de paquetages (package diagram) : permet de représenter la hiérarchie des modules du projet, leur organisation et leurs interdépendances. Cela simplifie les diagrammes, et les rend donc plus simple à comprendre.
- diagramme global d'interaction (interaction overview) : permet d'associer les notations du diagramme de séquence à celle du diagramme d'activité, ce qui permet de décrire une méthode complexe. C'est une variante du diagramme d'activité.
- diagramme de chronométrage (timing diagram) : permet de modéliser les contraintes d'interaction entre plusieurs objets, comme le changement d'état en réponse à un évènement extérieur.

Par ailleurs, le diagramme de collaboration est devenu "diagramme de communication" et la plupart des diagrammes ont été revus pour répondre aux nouveaux besoins (abstraction, automatisation...).

Cycle en V en gestion de projet : conception, réalisation, validation

- Le cycle en V en gestion de projet découle du modèle en cascade théorisé dans les années 1970, qui permet de représenter des processus de développement de manière linéaire et en phases successives.
- Ce mode de gestion de projet a été développé dans les années 1980 et appliqué au champ des projets industriels, puis étendu aux projets informatiques. Il a été remis en cause à partir du début des années 2000, sous l'effet de l'accélération des changements technologiques, favorisant davantage les méthodes dites « agiles ».
- La lettre V fait référence à la vision schématique de ce cycle, qui prend la forme d'un V : une phase descendante suivie d'une phase ascendante. Le cycle en V associe à chaque phase de réalisation une phase de validation.





Avantages de cette méthodologie en cycle V

- Le principal avantage du cycle en V est qu'il évite de revenir en arrière incessamment pour redéfinir les spécifications initiales.
- Chaque phase de conception demande la rédaction d'une documentation précise et exhaustive, où chaque point doit être validé par le produit final. Dès lors qu'une étape est validée, on ne revient pas en arrière et on passe à l'étape suivante sur une base solide; c'est la principale force du cycle en V.
- De par son aspect à la fois rigoureux et intuitif, le cycle en V demeure un processus facile à mettre en œuvre. Le travail préalable de définition des spécifications en début de projet fait que, une fois lancé, l'ensemble des étapes est connu des collaborateurs, qui peuvent se repérer facilement dans la temporalité du projet et connaître la finalité de leurs tâches. De la même manière, les documentations nécessaires à chaque étape sont réplicables d'un projet sur l'autre dans leur structure (cahiers des charges, cahiers de test...).
- En général, le cycle en V est plus adapté pour des réunions non quotidiennes, mais seulement des réunions de pilotage actant le passage d'une phase à l'autre. Son aspect linéaire autorise donc une organisation géographique éclatée, où le côtoiement des collaborateurs n'est pas clé dans le processus.

Modélisation Objet, le langage UML

Inconvénients cycle en V

- L'inconvénient principal du cycle en V se résume en deux mots : l'effet tunnel. Après une phase de définition précise du produit auquel doit l'équipe doit aboutir, le projet est lancé dans un « tunnel » constitué des phases évoquées plus haut. Mais que faire si les spécifications initiales sont dépassées ? Si le besoin du client vient à changer, ou a été mal exprimé ? Le cycle en V supporte donc mal les changements, ce qui est à la fois sa force et sa principale faiblesse.
- Il offre ainsi moins de réactivité par rapport au contexte technologique et économique, aux demandes du client, aux événements inopinés; la prise de risque s'en trouvera systématiquement limitée. L'effet tunnel est aussi induit par le travail conséquent de production de la documentation en début de projet, qui n'est plus rectifiable par la suite. Enfin, l'image du tunnel illustre le temps (parfois très) long qui sépare l'expression du besoin de la recette du produit final.

odélisation Objet, le langage UML (24)

Mise en œuvre du cycle en V

Pour quels projets?

Au regard des éléments exposés précédemment, les éléments suivants favorisent l'utilisation du cycle en V :

- Des exigences très précises émises par le client, par exemple dans le cadre d'un appel d'offres.
- La présence d'un prestataire, qui maîtrise l'ensemble des étapes de réalisation et requiert ainsi moins de communication entre les différents acteurs.
- La possibilité de suivre un cahier des charges inchangé du début à la fin, de par la nature du produit ou du projet.
- Un projet où l'environnement technologique évolue très peu, limitant ainsi les risques de décalage inhérents à l'effet tunnel.

lisation Objet, le langage UML

Une phase cruciale: la conception

- ◆ Les besoins du client doivent être recueillis de manière exhaustive et rigoureuse. Le dossier spécifications fonctionnelles détaillées (DSFD) doit faire l'objet d'un processus de validation suivi et définitif. Il doit synthétiser les demandes du client tout en couvrant l'ensemble du programme du projet.
- La description des moyens pour parvenir au produit final, quant à elle, ne doit intervenir qu'au stade des spécifications techniques (conception générale DAT: dossier d'architecture technique), de façon à éviter que les moyens ne définissent la fin!

élisation Objet, le langage UML

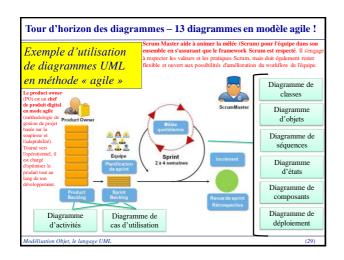
Qui valide les étapes de cycle V ?

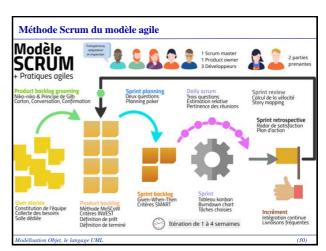
le cycle en V définit des étapes sans en définir les rôles ou les responsabilités. Il convient donc en début de projet de désigner les personnes ou les entités qui joueront le rôle de (par niveau de détail croissant):

- ♦ Maîtrise d'ouvrage (fonctionnel), ou MOA
- ♦ Maîtrise d'œuvre (système), ou MOE
- L'équipe architecturale, ou de conception générale (technique, métier)
- L'équipe de développement (par composant)
- Les rôles sont indiqués par séquence du cycle en V dans le schéma ci-dessus.

Modélisation Objet, le langage UML

Tour d'horizon des diagrammes - 13 diagrammes en cycle V! Diagramme de Exemple d'utilisation cas d'utilisation de diagrammes UML Expression des Diagramme besoins dans les phases d'un d'activités Analyse projet « classique » Diagramme de Conception composants Diagramme de classes Développement déploiement Test Diagramme de séquences Mise en Diagramme d'états service





Cycle en V vs. méthodes agiles

- De façon générale, l'on peut affirmer que le cycle en V se focalise sur le processus, tandis que les méthodes agiles privilégient le produit.
- Dans le cadre des méthodes agiles (Scrum, XP, RAD, ...), le projet s'affine par itérations, à travers la répétition d'un cycle d'opérations (le sprint dans le cadre de la méthode Scrum). Comme nous l'avons vu, le cycle en V définit l'intégralité du produit final dès les premières étapes, et ne laisse que peu de place à l'adaptation dans la suite du cycle.
- Ensuite, les méthodes agiles permettent d'élaborer le produit par incrémentation. On produit un peu plus à chaque fois, morceau par morceau, pour aboutir au résultat final. Le cycle en V concentre au contraire la réalisation de l'ensemble dans une seule phase, qui est intégralement conçue en amont et vérifiée en aval.
- Ce manque d'adaptation et de flexibilité du cycle en V a précisément conduit à l'émergence des méthodes agiles, en particulier dans le domaine du logiciel et du marketing, pour répondre aux changements de plus en plus rapides des technologies et des demandes des consommateurs.

Modélisation Objet, le langage UML

(31)

Qu'est-ce qu'une méthode agile ?

- ♦ Alors que les méthode traditionnelles visent à traiter les différentes phases d'un projet d'une manière séquentielle (que l'on nomme aussi cycle de développement en cascade ou encore cycle en V), le principe des méthodes Agiles est de le découper en sous-parties (ou sous-projets) autonomes (on parle également de développement itératif).
- Les parties (itérations) forment le projet dans sa globalité.

Modélisation Objet, le langage UML

(32)

Le Manifeste Agile, les principes fondateurs

Ces méthodes découlent du **Manifeste Agile**, des pratiques édictées par des experts en 2001 pour améliorer le développement de logiciels.

Les 4 valeurs mises en exergues :

- la primauté des personnes et des interactions sur les processus et outils.
- une préférence pour un logiciel fonctionnel plutôt qu'une documentation complète.
- une relation autre avec les clients : une collaboration permanente remplaçant une négociation contractuelle.
- une adaptation continue au changement et non le suivi rigide d'un plan.

Modélisation Obiet, le langage UML

(22)

Découlant de ces valeurs, le Manifeste agile définit 12 principes

- 1 La priorité n°1 est d'obtenir la satisfaction client au plus tôt par la livraison rapide et régulière de fonctionnalités attendues.
- 2 Accepter les demandes de changement en cours de projet. Ce sont des opportunités pour donner plus de valeur au projet et coller aux vrais besoins des clients.
- 3 Mettre en œuvre des livraisons rapides reposant sur des cycles courts (quelques semaines). Ces livrables doivent être opérationnels pour permettre des tests de validation des fonctionnalités attendues
- 4 Coopération forte et continue entre les utilisateurs et le développement. A 'inverse des méthode classiques où les rencontres entre les utilisateurs et la maîtrise d'oeuvre interviennent surtout en début et en fin de projet.
- 5 Donner de l'autonomie à des personnes impliquées et leur faire confiance.
- 6 Privilégier le face à face comme canal de communication entre les parties. Les interactions sont plus efficaces et plus riches. Tout va plus vite.
- 7 L'important est d'avoir une application opérationnelle.
- 8 Avancer avec un rythme constant compatible avec ce que peut produire l'ensemble des acteurs.
- 9 Focus sur la qualité technique et la qualité de conception pour construire une base solide renforçan l'agilité.
- 10 Rester simple dans les méthodes de travail : ne faire que ce qui est nécessaire.
- 11 Une équipe qui s'organise elle même produit de meilleurs résultats.
- 12 En revoyant régulièrement ses pratiques, l'équipe adapte son comportement et ses outils pour être plus efficace.

Modélisation Objet, le langage UML

(2.0)

Quels sont les avantages de la méthode agile ?

Cette approche permet d'obtenir :

- plus de flexibilité en travaillant sur des sous-parties autonomes. Elles peuvent être conçues, testée, modifiées de nouveau sans que l'ensemble du projet ne soit impacté. La prise en compte de besoins non identifiés dans la phase d'analyse ou bien l'émergence de nouvelles fonctionnalités au cours du développement peuvent être implémentées. Par expérience, il est difficile de penser à tout dans la phase de définition de besoin pour une approche classique de gestion de projet.
- Plus de fiabilité et de qualité: en simplifiant la complexité, en testant en continu, en favorisant les feedbacks, les échanges avec les clients.
- Des risques réduits : détection rapide grâce à des cycles courts.
- Une meilleure maîtrise des coûts: pas de coûteux retours en arrière
 si nécessaire le projet peut être stoppé rapidement.

Modélisation Objet, le langage UML

(35)

Mais aussi des limites de la méthode agile

- La flexibilité poussée à l'extrême peut conduire à un enlisement
- du projet. De nombreuses itérations sans que des directions ou décisions ne soient figées représentent un réel danger. L'une des causes possibles des revirements incessants des clients quant à leurs spécifications.
- ◆ Dans ces situations, le chef de projet (quelle que soit sa dénomination dans la méthode choisie) doit être capable d'arbitrer pour le bien du projet, mais également celui... du client.

Modélisation Objet, le langage UML

(36)

Les méthodes Agiles

Les principes de l'agilité sont repris d'une manière structurée par plusieurs méthodes. Focus sur l'une des plus populaires : **méthode Scrum**

 Cette méthode propose un cadre très structuré pour appliquer les principes de l'agilité.

Le Sprint, le coeur de Scrum :

 Cette approche repose sur des itérations de 2 à 4 semaines. Ce sont les fameux "Sprints". Il s'agit des sous-parties d'un projet comme le définit le principe Agile. Chaque Sprint a pour objectif de livrer au client une version potentiellement utilisable du produit.

Les Sprints successifs ajoutent des fonctionnalités au produit ou améliorent celles déjà développées. On parle d'incrément de produit.

 Un Sprint démarre lorsque le précédent est terminé. Il s'agit d'un processus incrémental.

Modélisation Obiet, le langage UML

(27)

Sprint

Le sprint repose sur 3 piliers que sont :

- la transparence : élaboration d'un standard commun
- pour permettre une compréhension partagée.
- l'inspection : des vérifications sont effectuées régulièrement.
- l'adaptation : en cas de dérive constatée lors de l'inspection, des ajustements sont décidés.

Modélisation Objet, le langage UML

(20)

Les Sprints se structurent autour de plusieurs outils organisationnels

- Sprint planning (Planification du sprint): réunion pour sélectionner et planifier les priorités de chaque Sprint en terme de liste des fonctionnalités produit (Sprint Backlog).
- Scrum (Mélée quotidienne): réunion journalière de coordination entre les membres de l'équipe projet. Elle prend fréquemment la forme de "Stand-up meeting" (réunion de courte durée, 10-15mn, tenue debout).
- Sprint Review (Revue de Sprint): réunion de synthèse à la fin de chaque Sprint afin de valider les fonctionnalités développées.
- Sprint Retrospective (Rétrospective de Sprint): venant immédiatement après la revue de Sprint, il s'agit d'un bilan dont l'objectif est l'amélioration continue des pratiques. L'équipe échange sur les réussites, les difficultés, relève ce qui a fonctionné ou non. Avec toujours des leçons à tirer pour les prochains Sprints

Modélisation Objet, le langage UMI

(39)

Comprenant des entrants et sortants du processus, appelés "artéfacts

- Product Backlog : liste des fonctionnalités du produit.
- Sprint Backlog: planification des éléments du Product Backlog à mettre en œuvre lors du Sprint pour livrer l'incrément de produit doté des fonctionnalités requises pour cette étape. Le Sprint Backlog n'est pas figé, mais est amené à évoluer durant le Sprint.
- L'incrément de produit : déjà évoqué plus haut.

Avec des rôles définis pour chacun :

- Product Owner PO (propriétaire du produit): l'expert métier, le maître d'ouvrage, représente le client et intervient sur le coté fonctionnel.
- Scrum Master (maître de mêlée): le coordinateur du projet et le garant du respect de la méthode Scrum.
- Team (équipe): les autres intervenants sur le projet (notamment les développeurs).

Modélisation Objet, le langage UML

(40)

Tour d'horizon des diagrammes - Restrictions & extensions

- <u>Restrictions</u>: au sein d'une organisation ou d'une équipe, on utilise un « sous-ensemble » de UML:
 - · Sous-ensemble de diagrammes,
 - · Sous-ensemble des possibilités offertes par chaque diagramme.
- <u>Extensions</u>: UML possède des mécanismes d'extension, qui permettent d'adapter le langage à une organisation, une équipe ou un domaine particulier.
- Un <u>profil</u> UML est un ensemble cohérent de concepts UML, d'extensions/restrictions, de contraintes, de règles et notations.
 - Exemples : profil EJB, profil SIG, profil RT, ...

Modélisation Obiet, le langage UML

(41)



Diagrammes communs à l'analyse et à la conception

Aspects statiques = structure du problème et de la solution

- Diagrammes de classes et d'objets
- Aspects dynamiques = comportement des éléments de la structure
- Diagrammes de séquence, de communications et d'états La conception est une phase charnière entre la spécification du problème et l'implantation de la solution. Il est donc important de garder la trace des décisions de conception : telle exigence extraite du cahier des charges et spécifiée dans l'analyse est traduite par tels et tels éléments du modèle de conception. C'est une des raisons qui explique que certains diagrammes sont utilisés dans le modèle de l'analyse et dans celui de la conception, la conception consistant à compléter ou raffiner le modèle de l'analyse avec des éléments techniques, par exemple comment rendre l'application accessible par un service Web sur l'intranet de l'entreprise.

Démarche d'analyse pour conception d'une architecture Objet

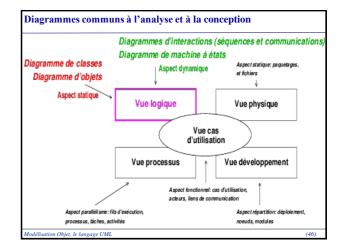
- L'analyse va être de plus en plus fine mais il faut une fois de plus éviter de plonger directement dans des solutions techniques connues tout en gardant à l'esprit les contraintes du contexte dans lequel l'application va être développée.
- Chaque cas d'utilisation contient un vocabulaire et des mécanismes que nous devons maintenant projeter à travers des entités informatiques.
- Il faudra durant notre analyse distinguer les classes de représentation des données qui seront dans la base de donnée (qui ne comportent que des attributs) et les classes qui vont constituer la partie logiciel du projet (qui comportent des attributs et des méthodes).

Modélisation Objet, le langage UML

(44)

Identification des classes candidates

- Il ne faut pas se tromper, les classes que nous allons identifier ne fournissent qu'une représentation statique du système. Cependant, nous devons garder à l'esprit que chaque classe va se composer d'un ensemble d'attributs et éventuellement d'opérations qui seront utilisés dans la descriptions dynamique du système.
- Chaque classe pourra être affinée par un diagramme d'état et/ou un diagramme d'activité qui va décrire son fonctionnement dynamique. Les diagrammes dynamiques vont utiliser les opérations des classes et vont aussi générer des opérations dans le cas des diagrammes de séquence.
- Enfin, durant la phase d'identification des classes il y a un certain nombre d'associations qui semblent nécessaires et il ne faut pas se priver de les créer.



Classes et objets

- Le diagramme de classes permet de représenter des classes, leurs propriétés et leurs relations avec d'autres classes.
- Le diagramme d'objets (des instances des classes) permet de représenter des objets, les valeurs de leurs propriétés et leurs relations avec d'autres objets.
- Mais au fait
 - Qu'est ce qu'un objet ?
 - Qu'est ce qu'une classe ?
 - Une classe peut-elle être un objet ? Et inversement ?

Modélisation Objet, le langage UML

(47)

Les objets

- Un objet est une entité identifiable du monde réel.
- Les objets informatiques définissent une représentation simplifiée des entités du monde réel.
- Un objet informatique est une structure de données valorisées qui répond à un ensemble de messages.
- Un objet peut représenter une entité concrète (personne, guitare, véhicule, ...) ou abstraite (gestionnaire de flux, ...).

Modélisation Objet, le langage UML

(48)

Les objets

- Une abstraction est un résumé, un condensé, une mise à l'écart des détails non pertinents dans un contexte donné.
- Mise en avant des caractéristiques essentielles et utiles.
- Dissimulation des détails (complexité).



odélisation Objet, le langage UML

Les objets

- ♦ L'état d'un objet :
 - regroupe les valeurs instantanées de tous les attributs d'un objet
 - · évolue au cours du temps
 - est la conséquence des comportements passés
- Exemples:
 - un signal électrique : l'amplitude, la pulsation, la phase, ...
 - une voiture : la marque, la puissance, la couleur, le nombre de places assises, ...
 - un étudiant : le nom, le prénom, la date de naissance, l'adresse,

Modélisation Objet, le langage UML

(50)

Les objets

- ◆ Le comportement
 - · décrit les actions et les réactions d'un objet
 - regroupe toutes les compétences d'un objet
 - se représente sous la forme d'opérations (méthodes)
- Un objet peut faire appel aux compétences d'un autre objet
- ♦ L'état et le comportement sont <u>liés</u>
 - Le comportement dépend souvent de l'état
 - L'état est souvent modifié par le comportement

Modélisation Obiet, le langage UML

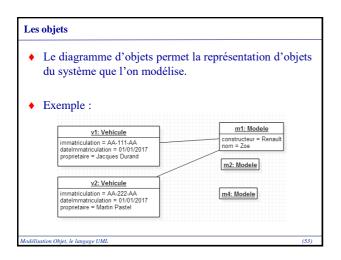
(51)

Les objets

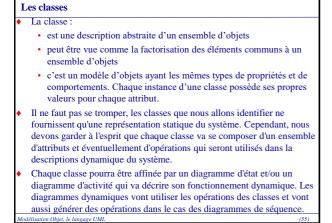
- Tout objet possède une identité qui lui est propre et qui le caractérise.
- L'identité permet de distinguer tout objet de façon non ambiguë, indépendamment de son état.

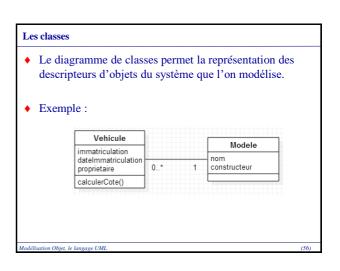
lodélisation Objet, le langage UML

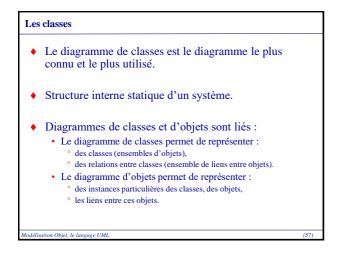
(52)

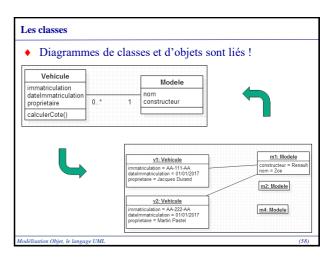


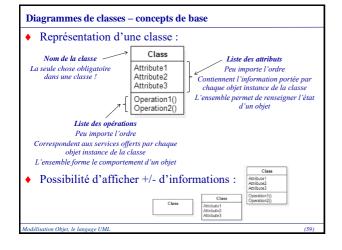


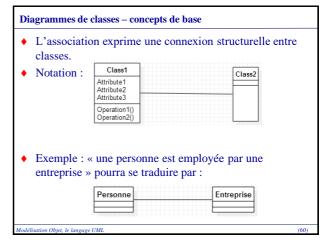


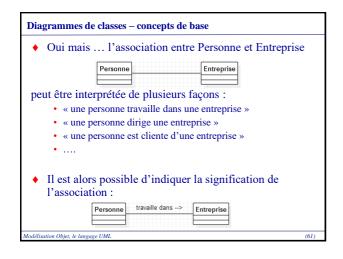


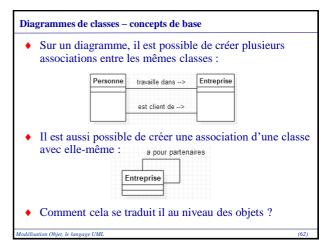


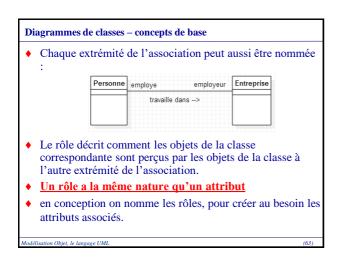


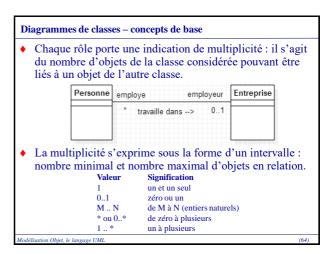


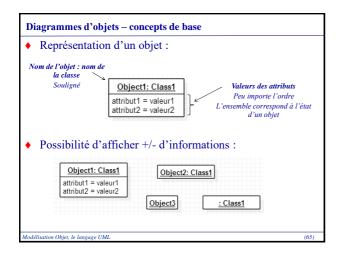


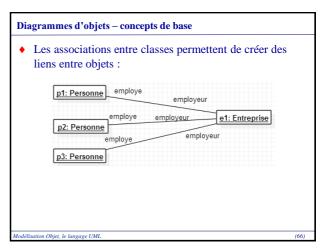


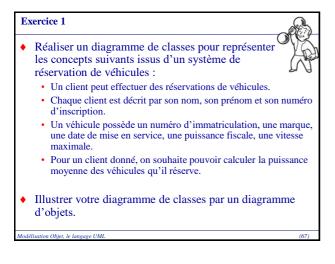








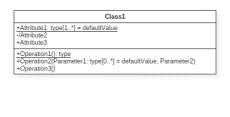






Propriétés complètes des attributs

La représentation complète d'une classe fait apparaître les attributs avec différentes caractéristiques en plus du nom (type, valeur par défaut, degré de visibilité, ...), et les opérations avec leur signature complète :



Propriétés complètes des attributs

- La forme complète de représentation d'un attribut est la suivante :
 - <visibilité> <nomAttribut> : <type> [borneInf..borneSup] =
 <valeur par défaut> {propriétés}
- Seul le nom est obligatoire!

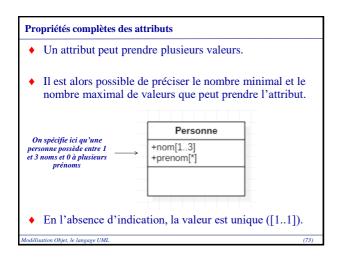
Aodélisation Objet, le langage UML

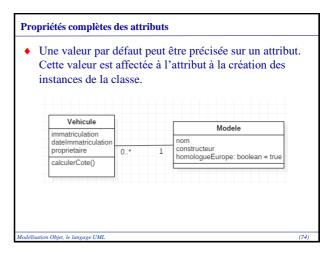
Propriétés complètes des attributs

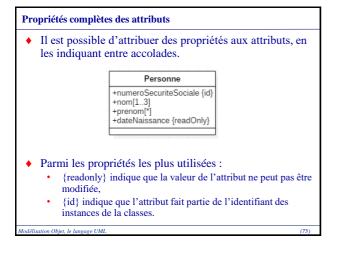
- Le symbole de visibilité correspond au concept objet d'encapsulation. Il représente le degré de protection de l'attribut:
 - + : publique (accessible à toutes les autres classes)
 - #: protégé (accessibles uniquement aux sous-classes)
 - ~: paquetage (accessible uniquement aux classes du paquetage)
 - : privé (inaccessible à tout objet hors de la classe)
- La visibilité peut être précisée sur chaque attribut et sur chaque opération.

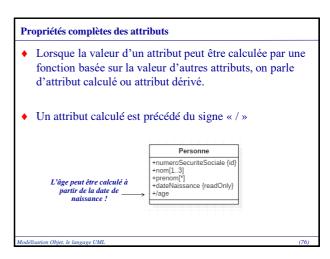
odélisation Obiet. le langage UML (71)

Propriétés complètes des attributs Le type permet de fixer l'ensemble des valeurs possibles que peut prendre un attribut. Il peut s'agir: d'un type standard: integer, string, boolean, real d'une classe: on préfèrera très souvent utiliser une association Préfèrez l'association et supprimez l'attribut « proprietaire » ! Vehicule immatriculation date/immatriculation object. Le language UML. Modélisation Objet. le language UML. (72)









Propriétés complètes des attributs

- Chaque instance d'une classe contient une valeur spécifique pour chacun de ses attributs. Dans certains cas cependant, il est utile de pouvoir définir des attributs dont la valeur est commune à l'ensemble des instances de la classe.
- On parle alors d'attribut statique ou attribut de classe, et on souligne l'attribut.

La vitesse maximale autorisée est la même pour toutes les instances de la classe ! Vehicule
+immatriculation
+datelimmatriculation
+proprietaire: findividu
+vitesseMaxAutorisee = 180
+dateMiseEnCirculation
+calculerCote()

odélisation Objet, le langage UML

Propriétés complètes des opérations

- La forme complète de représentation d'une opération est la suivante :
 - <visibilité> <nomOpération> (listeParamètres) : <typeRetour>
 [borneInf..borneSup] {propriétés}
- Comme pour les attributs, seul le nom, suivi des parenthèses (), est obligatoire!

Iodélisation Obiet, le langage UML

(78)

Propriétés complètes des opérations

- Certains concepts présentés pour les attributs s'appliquent aux opérations :
 - La visibilité de l'opération,
 - · L'intervalle pour indiquer le nombre de valeurs du retour,
 - Sur les paramètres : le type, l'intervalle pour préciser le nombre de valeurs autorisées et la valeur par défaut,
 - · Les propriétés.
- Il existe cependant des différences :
 - Lorsque l'opération renvoie un objet, il n'est pas possible de remplacer le typeRetour par une association!
 - Un paramètre peut être préfixé en indiquant sa direction : in, out ou inout.

Aodélisation Obiet, le langage UML

(79)

Propriétés complètes des opérations

 Il est également possible qu'une opération soit définie comme « statique » : cette opération est appelée sur la classe directement.

Vehicule

-immatriculation
-dateImmatriculation
-proprietaire: Individu
-proprietaire: Individu
-tyliesseMaxAutorisee = 180
-tateMiseEnCirculation
-calculerCote(dateCalcul: date = dateJour): integer
-tobtenirProprietaire(): Individu
+modifier\/itesseMaxAutorisee(nouvelleValeur: integer)

 Ces opérations ne peuvent pas manipuler d'attributs qui ne soient pas statiques!

Modélisation Objet, le langage UM

(80)

Classe d'association

La classe d'association est ... une classe! Cette classe permet de faire porter des informations aux liens entre instances de classes.

Commande +contient -> Produit prixCatalogue

DetailAchat quantite prixNegocie

- Une telle classe peut être dotée d'attributs, d'opérations, et de relations avec d'autres classes.
- Attention: une classe d'association ne peut être reliée qu'à une seule association, par contre elle peut être en relation avec d'autres classes.
- Faire exemple avec Achat et Article : où met on l'attribut quantité?

délisation Objet, le langage UML

Classe d'association

- Exemples d'utilisation de la classe d'association :
 - Des personnes empruntent des livres à la bibliothèque. Il est nécessaire de pouvoir retrouver la date de chaque emprunt.
 - Des personnes signent des accords entre eux. Il est nécessaire de conserver la date de ces accords et la ville où a eu lieu la signature.
 - Des personnes travaillent dans des entreprises. Pour chacun de leurs emplois, il est nécessaire de connaître le temps de travail et le type de contrat signé avec l'entreprise.

Modélisation Objet, le langage UML

(82)

Associations n-aire

- L'association permet de relier plus de deux classes. On parle d'association ternaire pour trois classes, ou plus généralement d'association n-aire.
- Représentation : un losange blanc ou au moyen d'une classe stéréotypée.
- Comment modéliser avec classes et relations le fait qu'un enseignant dispense un cours dans une salle donnée à un ensemble d'étudiants ?

Je veux connaître le gardien de but de chaque équipe, pour chaque gardien de but de chaque année.

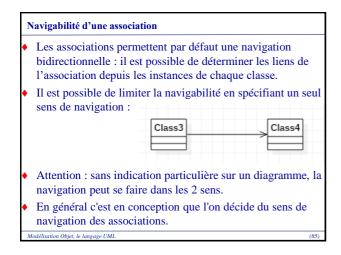
Pour chaque gardien de but de chaque équipe, je veux connaître chaque année le nombre de buts marqués, encaîssés, ...

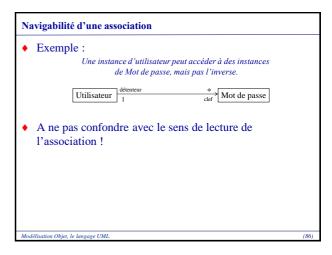
Associations n-aire

- Attention, il est souvent possible de modéliser un problème avec plusieurs associations binaires plutôt qu'avec une association n-aire.
- Exemple:
 - Un employé emprunte un véhicule de fonction pour se rendre sur un site
 - Un employé possède un véhicule de fonction. L'employé se rend sur des sites avec son véhicule de fonction

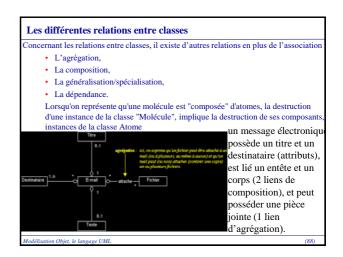
Modélisation Objet, le langage UML

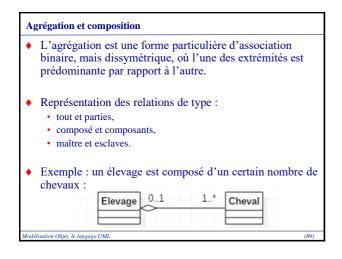
(04)

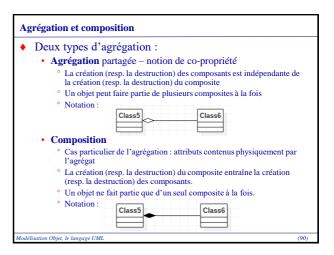


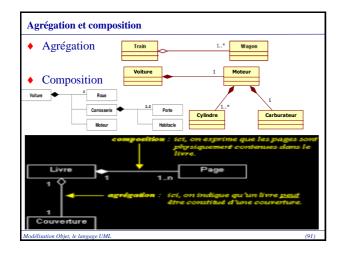


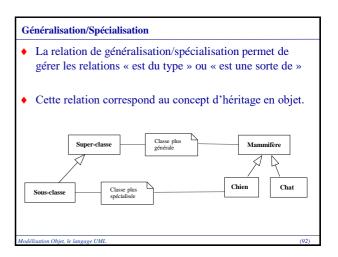
Les différentes relations entre classes Dépendance: un objet d'une classe travaille brièvement avec des objets d'une autre classe. Association: un objet d'une classe travaille avec des objets d'une autre classe pendant une durée prolongée. Agrégation: une classe détient et partage une référence à des objets d'une autre classe. Composition: une classe contient des objets d'une autre classe. Héritage: une classe est un type d'une autre classe. La composition est une forme particulière d'agrégation. La dépendance est une relation "un peu à part", souvent peu représentée, mais néanmoins importante! Modélisation Objet, le langage UML. (87)

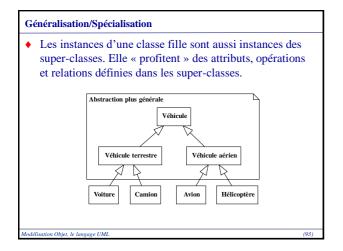


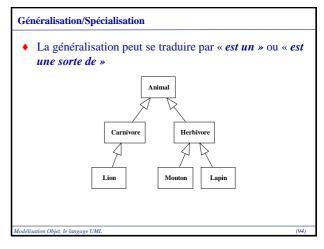


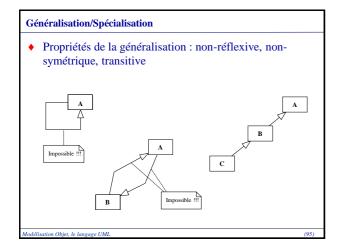


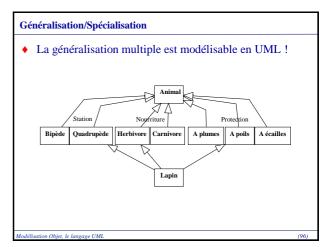


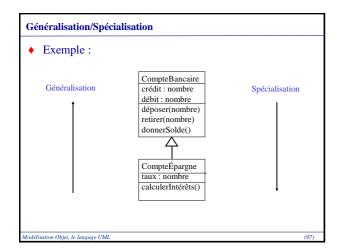


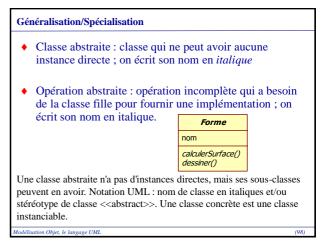


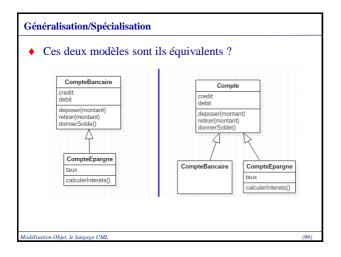


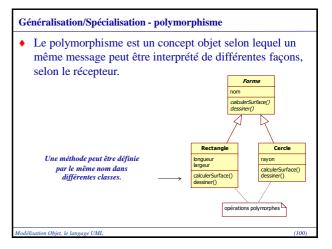














- L'interface est une classe totalement abstraite, sans attribut e dont toutes les opérations sont abstraites.
- Le concept d'interface permet la définition d'un contrat pour toutes les classes qui l'implémentent. La relation entre une interface et une classe qui implémente l'interface est appelée relation de réalisation.
- Notation :
- Une interface est une spécification des opérations qui sont visibles par l'environnement d'une classe. Conséquences : c'est une classe qui n'a pas d'implémentation (puisqu'il s'agit d'une spécification),

elle ne présente qu'une partie du comportement de la classe orrespondante (puisque cette dernière peut implémenter des méthodes internes").

Généralisation/Spécialisation - interfaces

- Une interface est une spécification des opérations qui sont visibles par l'environnement d'une classe. Conséquences :
- c'est une classe qui n'a pas d'implémentation (puisqu'il s'agit d'une spécification),
- elle ne présente qu'une partie du comportement de la classe correspondante (puisque cette dernière peut implémenter des méthodes "internes"),
- elle ne possède que des opérations (pas d'attributs, ni d'associations, ni d'états puisqu'elle n'est pas implémentée).
- Une interface est donc équivalente à une classe abstraite qui serait réduite à des opérations abstraites (elle ne possède ni attributs ni méthodes) et qui n'aurait pas de descendants (et donc pas d'instances).

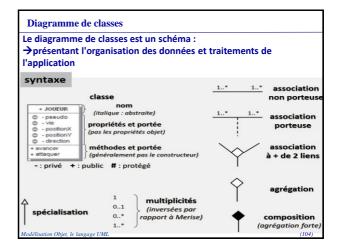
La relation de dépendance est une relation sémantique entre deux éléments selon laquelle un changement apporté à l'un peut affecter l'autre.

 Implique uniquement que des objets d'une classe peuvent fonctionner ensemble.

 Notation:

Une dépendance entre 2 classes stipule qu'une classe a besoin d'informations sur une autre classe pour pouvoir utiliser les objets de celle ci.

Indique que la modification de la cible peut avoir une incidence sur la source.



Exercice 2

Agrégation/Composition/Héritage/Dépendance: un pays à une capitale ? / une transaction boursière est un achat ou une vente ? / les fichiers contiennent des enregistrements ? / un polygone est composé d'un ensemble de points ordonné ? / une personne utilise un langage dans un projet ?/ les souris et les claviers sont des périphériques d'entrées sorties ? / des classes d'objets peuvent avoir plusieurs attributs ?

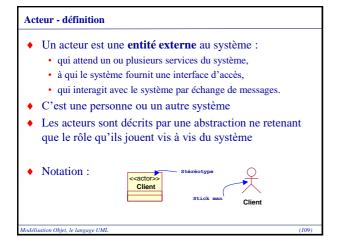
Réaliser un diagramme de classe pour chaque énoncé :

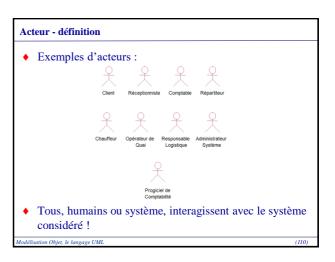
- Une voiture:
 - Ouatre roues, des pneus
 - ° Un moteur,
 - o Un coffre,
 - Oes passagers....
- Un chien: c'est un mammifère qui appartient à un propriétaire, possède quatre membres, donne naissance éventuellement à des chiots



Modèle de l'analyse Vue cas d'utilisation = ce que fait le système, les fonctionnalités Vue processus = ce que fait le système, les règles de gestion L'élément pivot du modèle du système informatique est la vue cas d'utilisation. Cette vue est la plus proche du client et des utilisateurs finaux. C'est elle que ces derniers comprennent le mieux. Elle est complétée par la vue processus qui décrit comment le système répond aux stimulus externes. Cette vue processus présente les processus métier, par exemple les règles de gestion d'un système d'information. La première vue est composée des disgrammes de cas d'utilisation alors que la seconde contient les diagrammes d'activité. Aspect dynamique : Interactions (séquence s. communications), machine a états Aspect strique : classes, obje to et paque tages Vue logique Vue développement Diagramme de cas d'utilisation Diagramme d'activités Vue processus Vue physique Aspect parallélisme: fils d'exécution, processus, étéche, activités Modélisation Objet, le langage UML (107)

Ce diagramme permet une description, en prenant le point de vue de l'utilisateur, du système à construire. Pas d'aspects techniques. Les deux concepts de base du diagramme : l'acteur, Le cas d'utilisation.

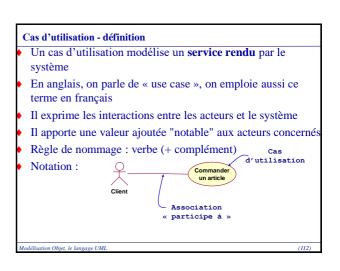




On ne doit pas raisonner en terme d'entité physique, mais en terme de rôle que l'entité physique joue Un acteur représente un rôle joué par un utilisateur qui interagit avec le système Exemple du monopoly : La même personne physique peut jouer le rôle de plusieurs acteurs (joueur, banquier) Plusieurs personnes peuvent également jouer le même rôle, et donc agir comme le même acteur (tous les joueurs) le responsable d'un magasin est parfois responsable du

magasin, parfois client, ... → il joue plusieurs rôles.

Acteurs vs Utilisateurs



Cas d'utilisation, enchaînements et scénarios

- Cas d'utilisation: représente un cas en général, une représentation générale et synthétique d'un ensemble de scénarios similaires décrits sous la forme d'enchaînements <u>Exemple</u>: un joueur joue un coup en lançant 2 dés
- Enchainement : succession d'étapes qui se réalisent lorsqu'un acteur déclenche un cas d'utilisation.
 - <u>Exemple</u>: un joueur joue un coup, les 2 dés ont une valeur identique, le joueur peut ensuite rejouer un autre coup
- Scénario: exécution d'un ou plusieurs enchaînements, joignant le début du cas d'utilisation à une fin normale ou non

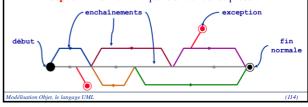
<u>Exemple</u>: le joueur Pierre joue : il obtient 7 avec les dés, se déplace rue de Belleville et achète la propriété

Modélisation Obiet, le langage UML

(112)

Cas d'utilisation vs scénario

- Un cas d'utilisation contient en général plusieurs enchainements.
- Pour décrire lisiblement un cas d'utilisation, on distinguera les trois types d'enchaînements :
 - Nominal : déroulement normal du cas d'utilisation
 - Alternatif : embranchements dans la séquence nominale
 - Exception: interviennent quand une erreur se produit



Pré et post conditions

- Pré-conditions : conditions obligatoires pour jouer un enchaînement du cas d'utilisation.
- Exemple : l'enchaînement « Faire un virement bancaire » a pour pré-condition : l'acteur doit être authentifié
- Post-conditions : changement intervenu dans le système considéré entre le début et la fin d'un enchaînement.
- Exemple : l'enchaînement « Faire une demande de prêt » a deux post-conditions :
 - · La création dans le système d'un nouveau dossier de prêt,
 - · L'envoi d'un mail de confirmation au client.

Modélisation Obiet, le langage UML

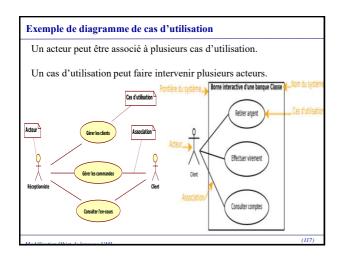
(115)

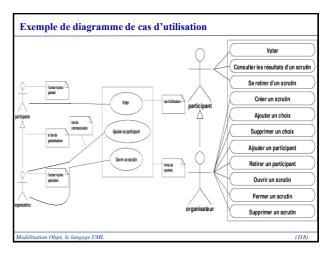
Intérêt des cas d'utilisation

- ◆ Un moyen de déterminer le **but** et le **périmètre** d'un système
- ◆ Utilisé par les utilisateurs finaux pour exprimer leur attentes et leur besoins → permet d'impliquer les utilisateurs dès les premiers stades du développement
- Support de communication entre les équipes et les clients
- Découpage du système global en grandes tâches qui pourront être réparties entre les équipes de développement
 - <u>UC (Use Case) Permet de concevoir les Interfaces Homme-</u> Machine
 - · Constitue une base pour les tests fonctionnels

Modélisation Objet, le langage UML

(116)

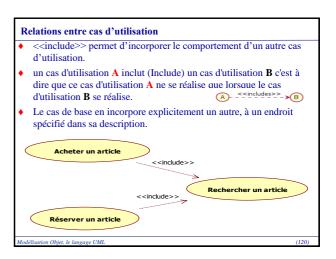


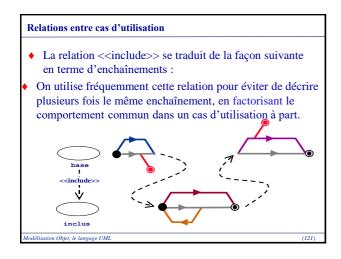


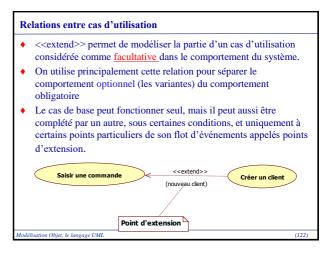
Relations entre cas d'utilisation

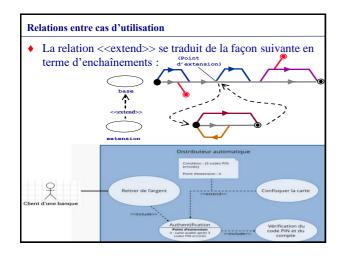
- UML définit trois types de relations standardisées entre cas d'utilisation :
 - une relation d'inclusion,
 - une relation d'extension,
 - une relation de généralisation/spécialisation.
- On peut également généraliser les acteurs

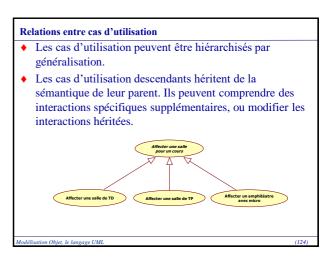
Modélisation Obiet, le langage UML (119)

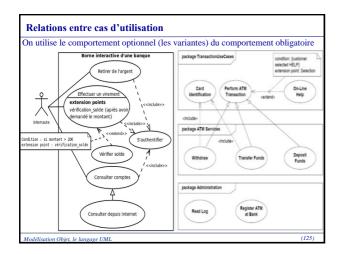


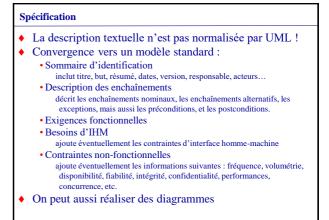










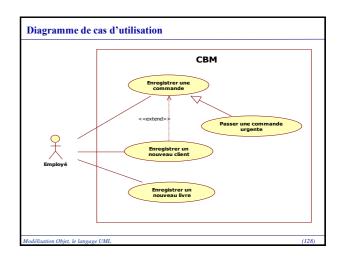


n Objet, le langage UML

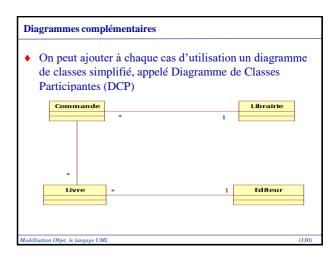
Exemple: La CBM

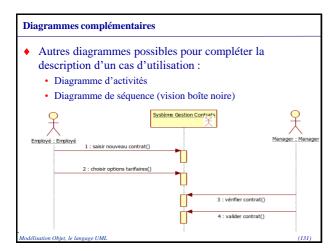
- La CBM (Computer Books by Mail) est une société de distribution d'ouvrages d'informatique qui agit comme intermédiaire entre les librairies et les éditeurs.
- Elle prend des commandes en provenance des libraires, s'approvisionne (à prix réduit) auprès des éditeurs concernés et livre ses clients à réception des ouvrages
- Il n'y a donc pas de stockage de livres.
- Seules les commandes des clients solvables sont prises en compte.
- Les commandes « urgentes » font l'objet d'un traitement particulier.
- La CBM désire mettre en place un Système Informatique lui permettant de gérer les libraires et les livres, et d'enregistrer les commandes.

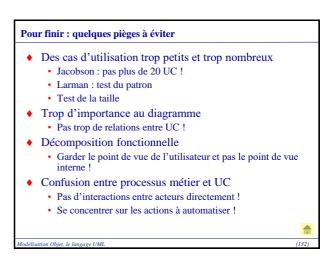
Modélisation Obiet, le laneage UML (127)

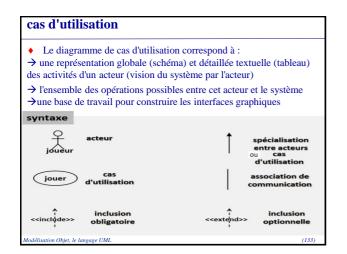


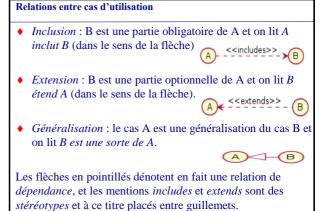
Spécification textuelle du cas « Enregistrer une commande » Acteur : l'employé de la coopérative Objectif: enregistrer une commande de livres Précondition : le libraire existe **Enchainement nominal:** 1 - l'employé sélectionne le libraire et vérifie sa solvabilité 2 - l'employé vérifie l'existence des livres 3 - l'employé précise la quantité pour chaque livre 4 – L'employé confirme la commande Postcondition : une nouvelle commande est créée $Enchainement\ d'exception\ 1:$ 1a - le libraire n'est pas solvable 1b - le système alerte l'employé et lui propose d'arrêter l'enregistrement Enchainement d'exception 2 : 2a - un des livres n'existe pas ${\bf 2b}-{\bf Le}$ système édite une lettre qui pourra être envoyée au libraire. La commande est placée en attente. odélisation Objet, le langage UML











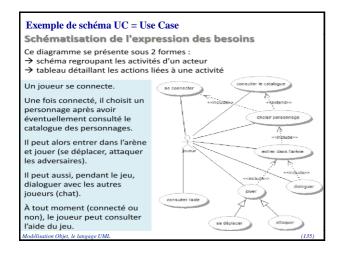


Tableau des opérations d'un cas d'utilisation	
Le tableau détaille les opérations d'un cas d'utilisation (un cercle). Il sert de base à la construction des interfaces.	
Cas d'utilisation	Se connecter
Acteur	joueur
Év. déclencheur	néant
Intérêts	Accéder au serveur afin d'être authentifié et accéder au jeu
Pré-conditions	Serveur actif
Post-conditions	Connexion établie
Scénario nominal	I'utilisateur saisit l'adresse IP du serveur I'utilisateur demande une connexion au serveur Ie serveur répond
Extensions	non
Contraintes	2a. L'adresse IP n'est pas remplie : aller en 1 3a. Le serveur retourne un message d'erreur : aller en 1 3b. Le serveur ne répond pas : aller en 1
Modélisation Objet, le langage UML (136)	

Description graphique des cas d'utilisation

Pour documenter les cas d'utilisation, la description textuelle est indispensable, car elle seule permet de communiquer facilement avec les utilisateurs et s'entendre sur la terminologie métier employée dans le cahier des charges ou l'expression des besoins.

En revanche, le texte présente des désavantages puisqu'il est difficile de montrer comment les enchaînements se succèdent, ou à quel moment les acteurs secondaires sont sollicités. En outre, la maintenance des évolutions s'avère souvent fastidieuse. Il est donc recommandé de compléter la description textuelle par un ou plusieurs diagrammes dynamiques UML.

Modélisation Objet, le langage UML

(137)

Description dynamique d'un cas d'utilisation

Pour les cas d'utilisation, on peut utiliser le diagramme d'activité car les utilisateurs le comprennent d'autant plus facilement qu'il paraît ressembler à organigramme traditionnel, ou un algorithme d'un programme.

Pour les scénarios particuliers, le diagramme de séquence est une bonne solution. Car il permet de démontrer les interactions entre l'acteur principal (positionné à gauche), puis un objet unique représentant le système à modéliser (boite noir) et des éventuels acteurs secondaires sollicités durant le scénario à droite du système.

Avec les intéressants ajouts au diagramme de séquence apportés par UML 2, en particulier les cadres d'interactions (avec les opérateurs loop, opt et alt par exemple), ainsi que la possibilité de référencer une interaction décrite par un autre Use Case.

baeusation Objet, ie langage UML

(138)

Exercice 3

Représenter par un diagramme de cas d'utilisation les éléments de l'énoncé suivant :

- Le système de déclaration des impôts en ligne permet aux contribuables :
 - De saisir toutes les informations relatives à leurs revenus de l'année précédente,
 - De mettre à jour leurs informations administratives (adresse postale, ...).
- La déclaration en ligne n'est accessible qu'aux contribuables authentifiés, au moyen de leur n° fiscal et de leur mot de passe.
- Lors de la saisie, le contribuable choisit de réaliser une déclaration simplifiée ou une déclaration complète.
- La déclaration d'impôts d'une année donnée n'est possible qu'après ouverture du service par les services fiscaux.

Modélisation Objet, le langage UML

(139)



Objectifs

- Décrire des scénarios particuliers
- Capturer l'ordre des interactions entre les parties du système (objets, composants, acteurs, ...)
- Décrire les interactions déclenchées lorsqu'un scénario d'un cas d'utilisation est exécuté
- Montrer la dynamique d'un système. On a dit qu'une application objet rendait des services grâce à la collaboration et à l'échange de messages entre objets, ici on va montrer ces échanges de messages.
- Rappel : scénario = instance d'enchaînement d'un UC.
- Montre les échanges de messages permettant de fournir les services attendus.

Modélisation Objet, le langage UML

(141)

Diagramme de séquence / diagramme de communication

- Ces diagrammes comportent :
 - des objets dans une situation donnée (instances)
 - les messages échangés entre les objets
- Les objets sont les instances des classes identifiées et décrites dans le diagramme de classes.
- ◆ On travaille sur un scénario → sur des instances particulières de classes.

Modélisation Objet, le langage UML

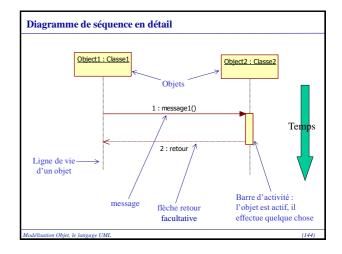
(142)

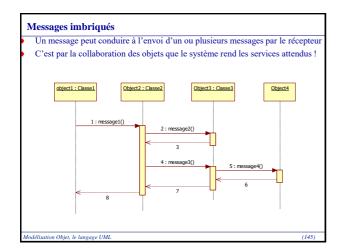
Diagramme de séquence / diagramme de communication

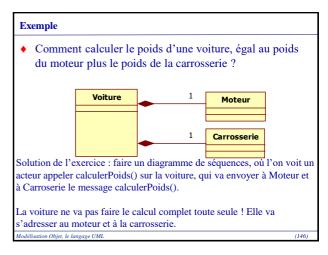
- UML propose deux types de diagrammes pour modéliser la collaboration entre les objets du système :
 - · Le diagramme de séquences,
 - Le diagramme de communication
- Ces deux diagrammes sont regroupés sous le terme de diagrammes d'interactions.
- Nous nous focaliserons dans un premier temps sur le diagramme de séquences.

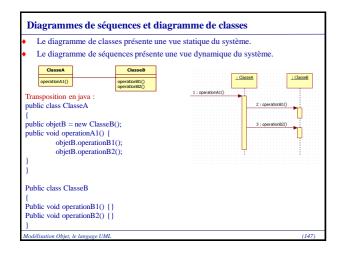
Modélisation Objet, le langage UML

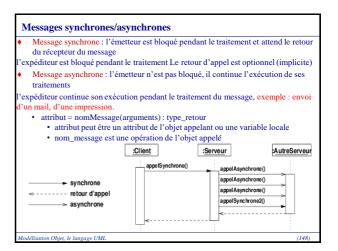
(143)











Syntaxe et types de messages

L'interaction la plus petite est l'événement. Un événement est une interaction pendant laquelle « quelque chose » arrive. Les événements sont les constituants de base des messages, aussi appelés « signaux » en automatique Un message est constitué d'un événement d'émission chez l'appelant et d'un événement de réception chez l'appelé, et possède une signature : « attribut = nom_message(arguments) : type_retour ». attribut peut être un attribut de l'objet appelant ou une variable locale. L'opération nom_message est une opération de l'objet appelé qui retourne un objet de type type_retour qui est affecté à l'attribut de l'objet appelant ou à la variable locale. La signature du message respecte la signature de l'opération appelée. La notation UML des diagrammes de séquence n'oblige pas à renseigner tous les éléments des prototypes des messages. Ainsi, les premiers diagrammes de séquence de l'analyse indiquent par exemple uniquement les noms des opérations. Ensuite, les mêmes diagrammes sont raffinés pour y ajouter les arguments et les types de retour, puis les attributs des classes appelantes ou variables locales recevant les valeurs de retour.

Syntaxe et types de messages

Un message synchrone est une invocation d'opération bloquant l'appelant jusqu'à ce que l'appelé effectue le traitement et retourne l'appel, que ce dernier contienne ou non une valeur de retour.

Un retour d'appel est un message que vous pouvez représenter à la fin de la barre d'activation de l'appelé pour indiquer que le traitement est terminé et que la valeur de retour est passée à l'appelant qui peut reprendre son traitement après la fin de l'appel synchrone. Comme la barre d'activation, la représentation de l'appel de retour est optionnelle.

Un message asynchrone est initié par l'appelant qui continue son exécution car il n'attend pas le retour de l'appel. L'appelant peut par exemple émettre à destination d'un même objet appelé une salve de messages asynchrones avant un message synchrone demandant le résultat de tous les traitements précédents. Cela lui permet d'effectuer aussi des traitements en parallèle, et dans cet exemple, cela diminue d'autant le nombre de messages échangés (car il n'y a qu'un seul retour d'appel). Il est important de noter qu'un message asynchrone ne possède pas de valeur de retour.

Modélisation Objet, le langage UML

(150)

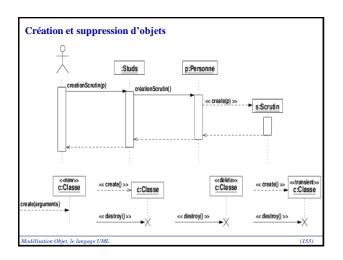
Création et suppression d'objets

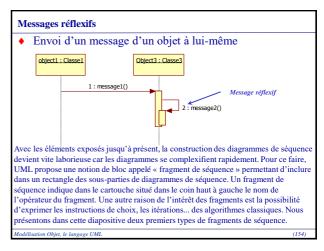
Certains objets vivent pendant tout le diagramme, d'autres sont créés et/ou meurent pendant la séquence. Pour montrer qu'un participant est créé lors de la séquence, vous pouvez soit placer l'objet en haut du diagramme en y ajoutant le stéréotype «new» et utiliser un message synchrone appelant l'opération create(arguments), soit placer l'objet plus bas dans le diagramme au niveau du message synchrone de création et utiliser le stéréotype «create(arguments)» pour nommer ce message synchrone. Le message correspondant à l'opération de création d'un objet est un message particulier. L'opération est dans la suite nommée un constructeur. Notez que l'objet en question n'existe pas avant le message, c'est-à-dire avant sa création, et que l'opération utilisée ne possède pas de type de retour ; c'est une seconde particularité.

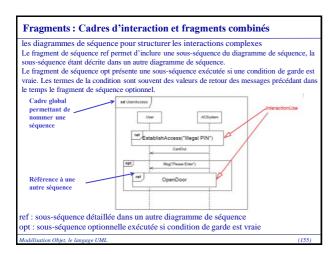
Par analogie avec la création d'un objet dans un diagramme de séquence, la destruction d'un objet pendant une séquence est modélisée soit en plaçant l'objet en haut du diagramme en y ajoutant le stéréotype «delete», la destruction étant repérée par un message synchrone appelant l'opération destroy(), soit en plaçant l'objet plus bas dans le diagramme au niveau du message synchrone de suppression. Enfin, un objet est dit transitoire (en anglais, transient) lorsqu'il est créé puis détruit durant la même séquence. Le stéréotype de l'objet est alors « «transient» ».

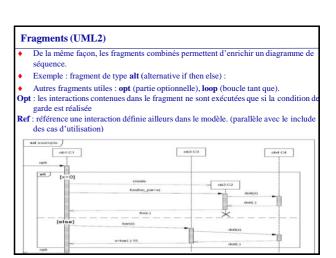
Modélisation Objet, le langage UML

(152)

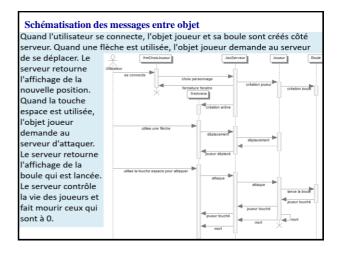


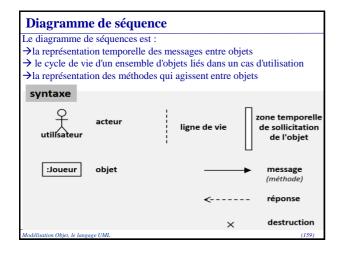






◆ Dans la vue fonctionnelle du système ◆ Diagramme complémentaire de description d'un cas d'utilisation ◆ Décrire un scénario d'un cas d'utilisation : décrire les interactions entre le système et son environnement (vision boîte noire) ◆ Appelé diagramme de séquence « système » ◆ Dans l'analyse détaillée et la conception OBJET du système ◆ Décrire les interactions internes du système : les interactions entre les objets Modélisation Objet, le langage UML





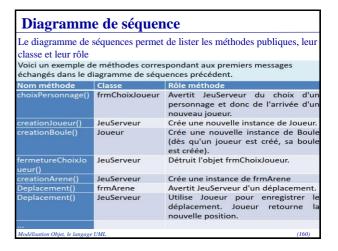
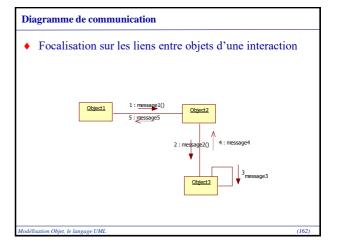
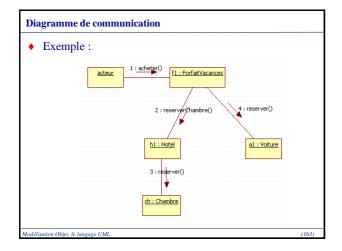


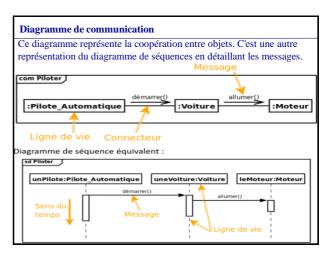
Diagramme de communication

- Diagramme de séquences : l'accent est mis sur l'ordre temporel des interactions
- Diagramme de communication: l'accent est mis sur l'examen des interactions vis-à-vis des liens entre objets.
- Équivalents en UML1, quelques nouveautés sur le diagramme de séquences en UML2, non disponibles sur le diagramme de communication
- Les diagrammes de séquence et de communication sont tellement similaires que de nombreux outils de modélisation permettent de transformer l'un en l'autre et vice-versa.

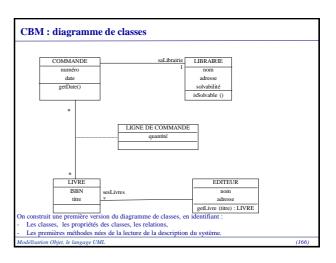
Modélisation Objet, le langage UML (10







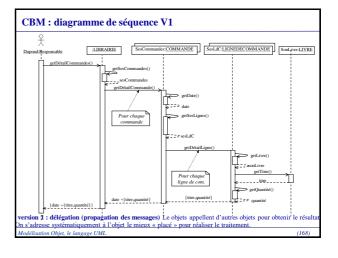
Description du système : Reprendre l'énoncé de l'exercice sur les cas d'utilisation. Objectif : Réaliser un diagramme de classes pour le système CBM, Réaliser un diagramme de séquence objet lorsque qu'un utilisateur demande les détails de toutes les commandes d'une librairie donnée. Modélisation Objet, le langage UML

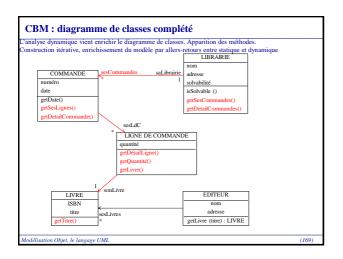


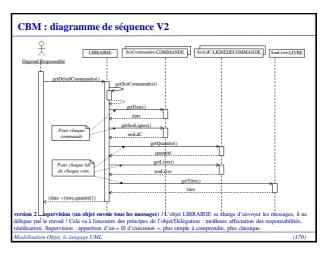
CBM : un diagramme de séquence

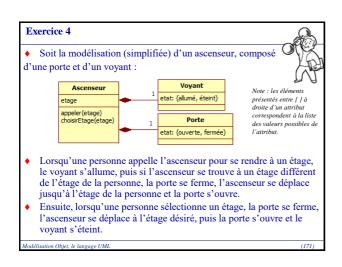
- Diagramme de séquence : « Communiquer les détails de toutes les commandes d'une librairie donnée »
- Ce diagramme correspond à la méthode getDétailCommandes() de la classe LIBRAIRIE
- On fait l'analyse/conception de la méthode getDétailsCommande()
- Cela semble intéressant car a priori met en jeu de nombreuses interactions entre objets.

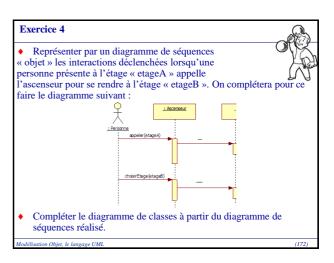
Andélisation Objet, le laneage UML (16



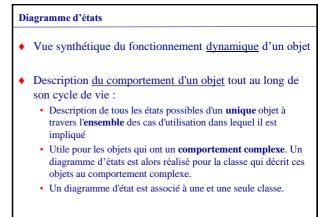


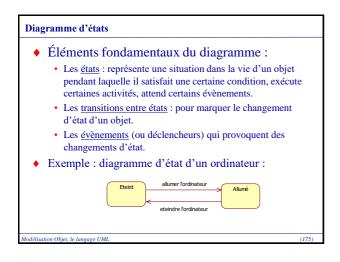


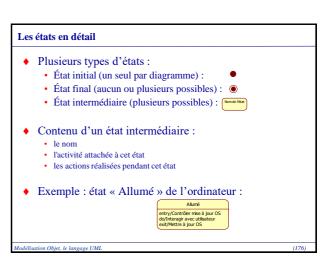


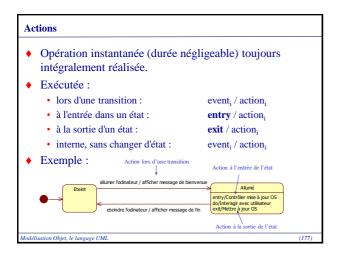


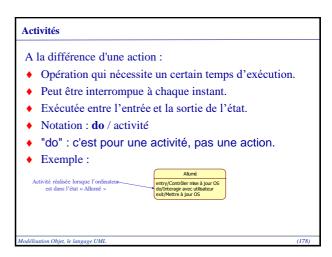




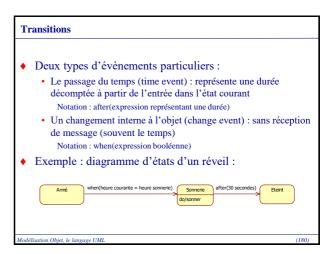


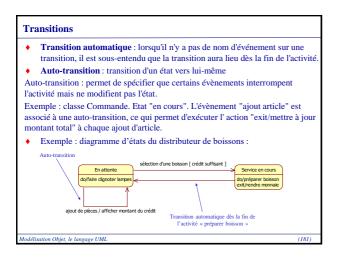




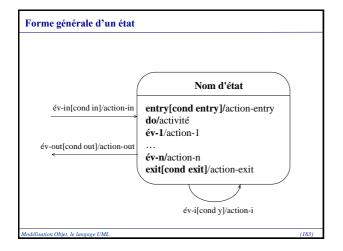


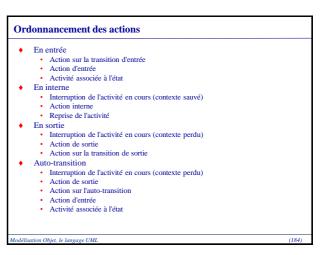
Transitions Passage unidirectionnel et instantané d'un état à un autre état Déclenchée: par un événement, automatiquement à la fin d'une activité (transition automatique). Condition de garde: condition booléenne qui autorise ou bloque la transition Action: réalisée lors du changement d'état Syntaxe complète: <Événement> [<Garde>] / <Action> Exemple: Aluné Eteirt Aluné eteindre fodrateur / afficher message de bienvenue eteindre fodrateur / afficher message de bienvenue eteindre fodrateur / afficher message de fin

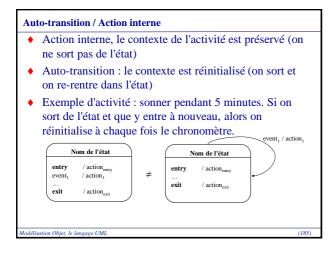


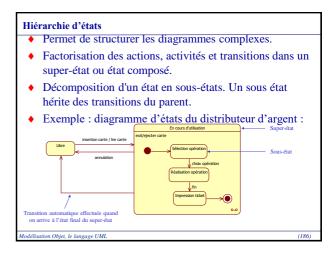


Représenter sous forme de diagramme d'états le fonctionnement d'un lecteur de DVD, dont voici la description: Le lecteur possède un tiroir qui peut recevoir plusieurs disques. A la fin de la lecture d'un disque, le lecteur démarre la lecture du disque suivant. A la fin du dernier disque, le lecteur s'arrête. L'utilisateur peut lancer la lecture des disques, arrêter la lecture ou mettre en pause le lecteur.

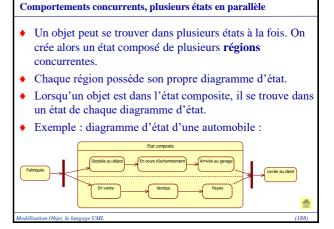


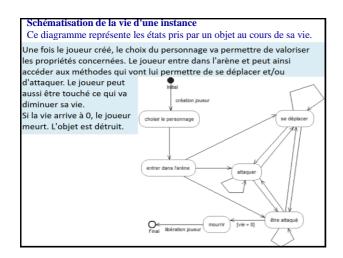


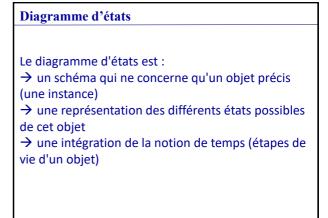




Pseudo-état historique • Quand une transition sort d'un super-état: • le dernier sous-état atteint n'est plus connu, • une nouvelle entrée dans le super état redémarre au premier sous-état. • Un pseudo-état historique permet de mémoriser le dernier sous-état. • Notation: • Exemple: diagramme d'états d'un développement informatique: • Transité de l'agramme d'états d'un développement informatique:







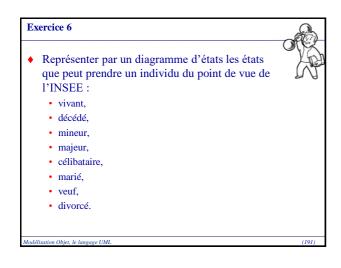




Diagramme d'activités

- Permet la modélisation dynamique d'un système.
- Mettre l'accent sur les enchaînements et les conditions pour exécuter et coordonner des actions.
- Utilisation :
 - · Décrire un processus métier,
 - Décrire un cas d'utilisation,
 - Décrire un algorithme ou une méthode.

Modélisation Obiet, le langage UML

(193)

Éléments de base du diagramme

- Une action modélise une étape dans l'exécution d'un flot (algorithme ou processus).
- Les actions sont reliées par des transitions, en général automatiques.
- Exemple:



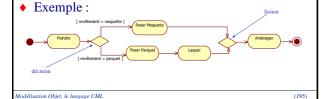
• Noter la similitude avec le diagramme d'états!

odélisation Objet, le langage UML

(194)

Décisions et fusions

- Une décision modélise un choix entre plusieurs flots
- Une **fusion** rassemble plusieurs flots alternatifs. Équivalent d'un **OU** dans un texte.
- ♦ Notation : ◆



Bifurcation et union

- Une bifurcation (ou débranchement) modélise une séparation d'un flot en plusieurs flots concurrents.
- Une union (ou jointure) synchronise des flots multiples.

Équivalent d'un ET dans un texte.

Notation: ou

Exemple:

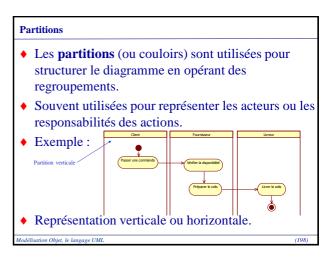
Resemble is ingrédients

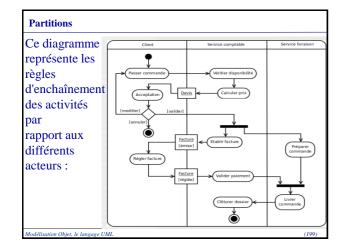
Préchauffer le four

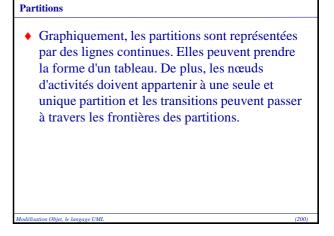
bifurcation

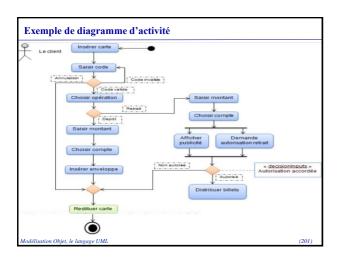
49

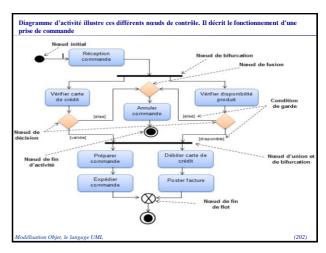
Objets Il est possible de faire apparaître des objets dans le diagramme. L'état de l'objet peut être indiqué si l'objet change d'état durant l'exécution du diagramme. Permet de montrer la circulation et l'utilisation d'objets au sein de l'activité. Exemple: Modélisation Objet. le langage UML. (197)

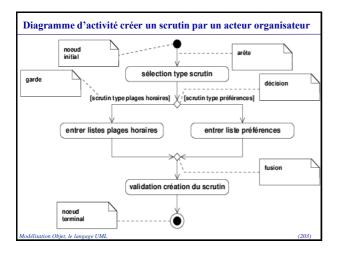


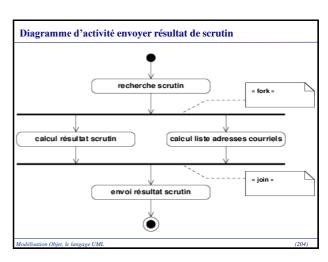












Nœud initial / Nœud final

Nœud initial

C'est un nœud de contrôle à partir duquel le flot débute lorsque l'activité enveloppante est invoquée. Une activité peut avoir plusieurs nœuds initiaux et un nœud a un arc sortant et pas d'arc entrant. Il est représenté par un petit cercle pleir

Nœud final

Un nœud final est un nœud de contrôle possédant un ou plusieurs arcs entrants et aucun arc sortant. Il y a deux type de nœuds finaux: nœud de fin d'activité et nœud de fin de flot. Lorsque l'un des arcs d'un nœud de fin d'activité est activé, 'exécution de l'activité enveloppante s'arrête et tous les nœuds ou flots actifs appartenant de cette activité est abandonné. Si l'activité a été appelé par un appel synchrone, un message Reply qui contient les valeurs sortantes est transféré en retour à l'appelant. Un nœud de fin d'activité est représenté par un cercle contenant un petit cercle plein,

Un flot est terminé lorsqu'un flot d'exécution atteint un nœud de fin de flot. Mais cette fin de flot n'a pas d'incidence sur les autres flots actifs de l'activité enveloppante. Un nœud de fin de flot est représenté par un cercle vide barré d'une

Nœud d'union

C'est un nœud de contrôle qui synchronise des flots multiples. Il possède plusieurs arcs entrants et un seul arc sortant. Lorsque tous les arcs entrants sont activés, l'arc sortant l'est aussi également. Un nœud d'union est aussi représenté par un trait plein épais. (cf. au diagramme suivant. Enfin il est possible de fusionner un nœud de bifurcation et un nœud d'union, possédant donc plusieurs arcs entrants et sortants.

Modélisation Objet, le langage UML

(206)

Nœud de bifurcation

Un nœud de bifurcation est un nœud de contrôle qui sépare un flot ou plusieurs flots concurrents. Il possède donc un arc entrant ou plusieurs arcs sortants. Il est souvent accordé avec un nœud d'union pour équilibrer la concurrence

Modélisation Objet, le langage UML

(207)

Nœud de fusion

Un nœud de fusion est un nœud de contrôle rassemblant plusieurs flots alternatifs entrants en un seul flot sortant. On ne l'utilise pas pour synchroniser des flots concurrents mais pour accepter un flot parmi plusieurs. Un nœud de fusion est représenté par un losange comme le nœud de décision.

Graphiquement, il est possible de fusionner un nœud de fusion et un nœud de décision, c'est-à-dire de posséder plusieurs arcs entrants et sortants. Il est aussi possible de fusionner un nœud de décision ou de fusion avec un autre nœud. Mais pour mieux mettre en évidence un branchement conditionnel, il est préférable d'utiliser un nœud de décision.

Modélisation Objet, le langage UML

(208)

Nœud de décision

C'est un nœud de contrôle qui permet de faire un choix entre plusieurs flots sortants. Il possède un arc entrant et plusieurs arcs sortants, accompagnés de conditions de garde pour conditionner le choix. Quand le nœud de décision est atteint et qu'aucun arc en aval n'est franchissable (ce qui veut dire qu'aucune condition est vraie), cela signifie que le modèle est mal formé. C'est pour cela que l'utilisation d'une garde (else) est recommandée après un nœud de décision, car elle garantit un modèle bien formé. En effet, la condition de garde est validée si et seulement si toutes les autres gardes des transitions ayant la même source sont fausses. Lorsque plusieurs arcs sont franchissables (c'est-à-dire que plusieurs conditions de garde sont vraies), l'un d'entre eux est retenu et ce choix est non déterministe. Un nœud de décision est représenté par un losange.

Modélisation Objet, le langage UML

(209)

Signaux

- Les signaux permettent de représenter des interactions avec des participants externes (acteurs, systèmes, autres activités,...).
- Un signal reçu déclenche une action du diagramme. Un signal émis est un signal envoyé à un participant externe.
- Exemple : interactions entre 2 diagrammes d'activités:



Utilisation du diagramme d'activité

- Le diagramme d'activité est plus facile à lire et à comprendre que les autres diagrammes. Il permet donc de communiquer avec des acteurs « non techniques ».
- Il s'utilise à différents niveaux :
 - [1] Modélisation d'un processus métier (BPM),
 - [2] Modélisation d'un cas d'utilisation,
 - [3] Modélisation du comportement interne d'une méthode (algorithme).



Modélisation Objet, le langage UMI

Exercice 7 • Réalis

 Réaliser un diagramme d'activités pour décrire le passage au guichet d'enregistrement d'un vol dans un aéroport.



- Elle vérifie alors que le billet correspond bien au vol en cours d'enregistrement, et la correspondance entre le nom écrit sur le billet et le nom écrit sur la PI.
- En cas de problème, le passager peut être réorienté vers l'agence de voyage qui se trouve dans l'aéroport.
- Si tout est correct, l'hôtesse demande les préférences au passager (placement dans l'avion, ...).
- Puis, si le passager est muni de bagages, elle prend les bagages et édite un reçu à destination du passager.
- 6. En parallèle, la carte d'enregistrement est imprimée.
- Enfin la carte d'enregistrement est remise au passager, ainsi qu'une documentation sur la compagnie aérienne.

Modélisation Objet, le langage UML

(212)



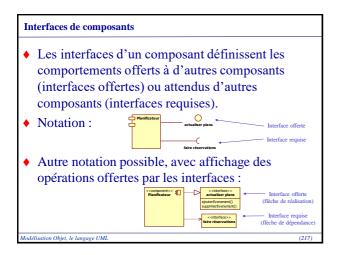


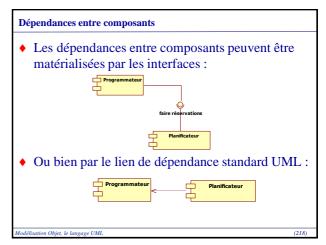
Diagramme de composants

- Le diagramme de composants présente l'architecture applicative statique du système, alors que le diagramme de classes présente la structure logique du système.
- Il permet de montrer les **composants** du système et leurs **dépendances** dans leur environnement d'implémentation.
- Les composants permettent d'organiser un système en « morceaux » logiciels.
- Ce diagramme représente l'organisation des ressources <u>au niveau logique.</u> C'est la représentation des unités (bases de données, fichiers, librairies...) dont l'assemblage compose l'application

odélisation Obiet, le langage UML (2)

Un composant est un élément encapsulé, réutilisable et remplaçable du logiciel. Ce sont des « briques de construction », qui permettent en les combinant de réaliser un système. Notation : ☐ composant ou composant ☐ Exemples : enregistreur d'évènements, éditeur PDF, convertisseur de format, ...





Conception d'un composant

- Un composant doit fournir des services bien précis, qui doivent être cohérents et génériques pour être réutilisables dans différents contextes.
- Le comportement interne, réalisé par un ensemble de classes, est masqué aux autres composants; seules les interfaces sont visibles.
- On peut substituer un composant à un autre si les interfaces sont identiques.
- Le « découpage » d'un système en composants se fait en recherchant les éléments qui sont employés fréquemment dans le système.

Andélisation Objet, le langage UML

Vue boîte noire: montre l'aspect extérieur d'un composant: interfaces fournies et requises, liens avec d'autres composants.

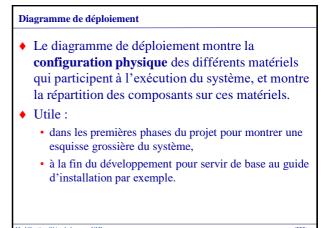
- → pour se concentrer sur les problèmes d'architecture applicative générale du système.
- ◆ Vue boîte blanche : montre les classes, les interfaces et les autres composants inclus.
 - → Pour montrer comment un composant rend ses services au travers des classes qu'il utilise.

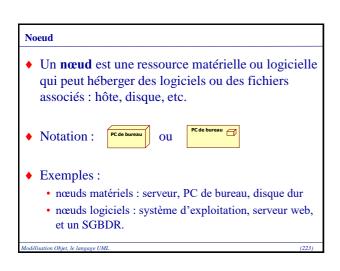
odélisation Objet, le langage UML

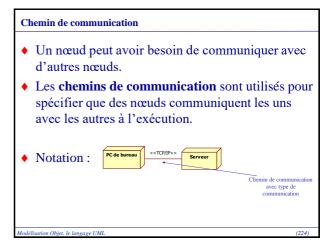
Vues d'un composant

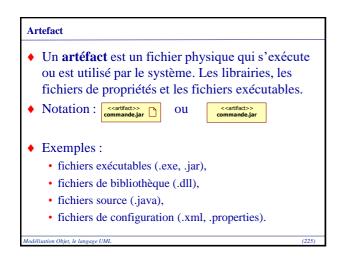


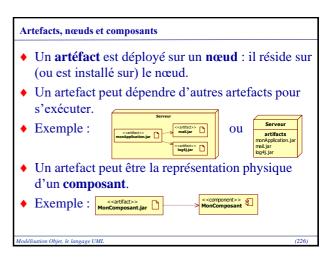


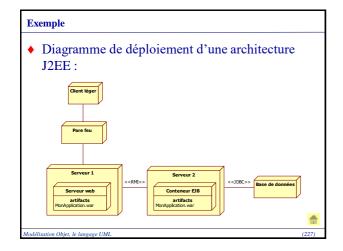


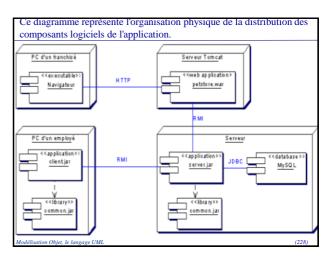


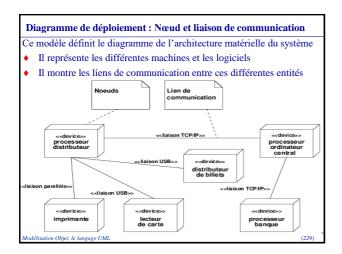


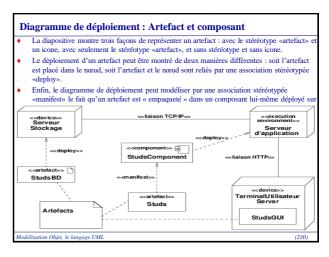










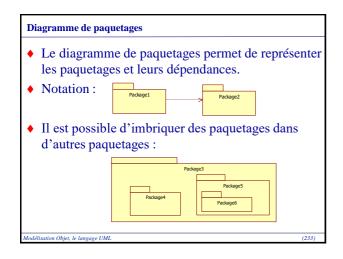


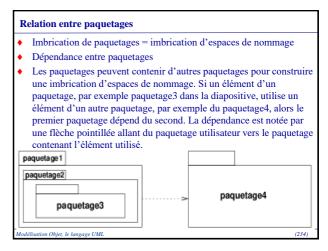


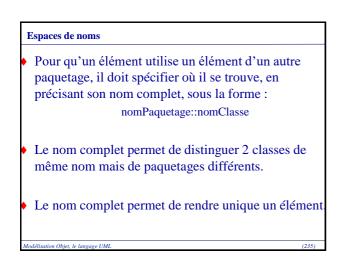
Paquetage

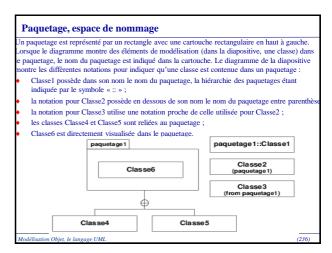
- Problème: un système peut contenir plusieurs centaines de classes ou d'éléments de modélisation. Comment organiser ces classes et ces diagrammes?
- Solution : rassembler les classes dans des groupes logiques.
- Le paquetage est un regroupement d'éléments UML, par exemple un regroupement de classes et d'autres diagrammes.
- Un paquetage peut aussi être un regroupement d'autres éléments comme les cas d'utilisation.

Objet, le langage UML (232)









Visibilité d'un élément

- Les éléments peuvent avoir une visibilité publique ou privée:
- Visibilité publique : visible et accessible de l'extérieur du paquetage.
- ♦ Notation : +
- Visibilité privée : non visible et non disponible de l'extérieur du paquetage.
- ♦ Notation : -

Modélisation Obiet, le langage UMI

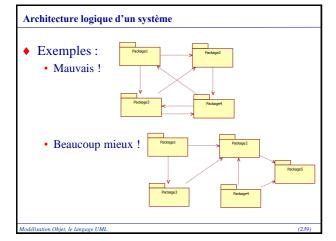
(227)

Architecture logique d'un système

- Les dépendances entre paquetages sont issues des dépendances entre éléments des paquetages.
- De trop nombreuses dépendances entre paquetages conduisent à un système fragile et peu évolutif.
- ◆ Objectif de l'**architecture** : limiter les dépendances entre paquetages.
- Si une classe A du paquetage 1 dépend d'une classe B du paquetage 2, alors le paquetage 1 dépend du paquetage 2.

Iodélisation Objet, le langage UML

(238)

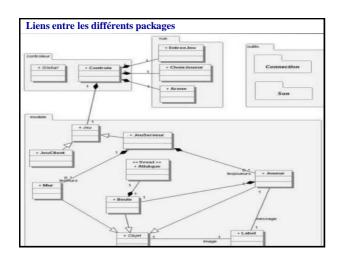


Les conseils de l'architecte

- Faire des regroupements en assurant une cohésion forte à l'intérieur des paquetages et un couplage faible entre les paquetages.
- Éviter les dépendances circulaires.
- Aller dans le sens de la stabilité.

odélisation Objet, le langage UML

(240)



Conclusion

- Modèle pour appréhender la réalisation d'un système informatique
- Diagrammes UML qui permettent d'aider à la réflexion, à la discussion (clients, architectes, développeurs, etc.)
- Pas de solution unique mais un ensemble de solutions plus ou moins acceptables suivant les contraintes du client, et les logiciels et matériels disponibles
- Une solution acceptable est obtenue par itérations successives

Modélisation Obiet, le langage UMI.

(242)

Exercice de révision sur les cas d'utilisation

La Direction des Ressources Humaines (DRH) d'une grande entreprise a décidé de mettre en place un système informatique pour gérer les demandes de mobilité de son personnel dans les différentes agences de l'entreprise.

Avec le futur système, les employés pourront renseigner leurs informations personnelles (qualifications, langues pratiquées, diplômes avec date d'obtention et mention), et faire des demandes de mobilité.

Faire une demande de mobilité consiste à indiquer les zones géographiques souhaitées et la période de validité de la demande.

Lorsqu'un employé fait une demande de mobilité, il doit avoir mis à jour ses informations personnelles depuis moins de 3 mois, sinon le système lui demandera de les mettre à jour. D'autre part, si l'employé a déjà une demande de mobilité en cours, alors le système doit l'avertir et lui demander s'il désire annuler ou non sa demande en cours avant de saisir sa nouvelle demande.

Chaque demande de mobilité est traitée par le directeur de l'agence de l'employé, qui valide ou refuse la demande. S'il la valide, la demande est traitée par un employé de la DRH qui recherche et sélectionne une ou plusieurs agences susceptibles d'accueillir l'employé. Chaque directeur d'agence sélectionnée doit alors, dans un délai d'une semaine, étudier le profil de l'employé et accepter ou refuser l'employé pour son agence.

odélisation Objet, le langage UML

Exercice de révision sur les cas d'utilisation

Lorsque plusieurs agences acceptent un employé, c'est au responsable de la DRH de traiter la demande et sélectionner l'agence dans laquelle l'employé sera affecté.

Un directeur d'agence peut renseigner les profils recherchés pour son agence : indiquer les qualifications recherchées, et pour chacune d'elles le niveau d'expertise (débutant, confirmé, expert).

- 1. Réaliser un diagramme de cas d'utilisation présentant les acteurs et les cas d'utilisation du système
- 2. Décrire le cas d'utilisation de la demande de mobilité par l'employé

odélisation Objet, le langage UML

