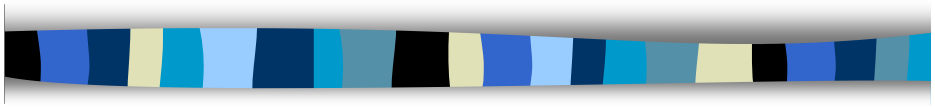
UML : Unified Modeling Language



Mounia Fredj
mounia.fredj@um5.ac.ma

Bibliographie / Webographie

B. Charroux, A. Osmani, Y. Thierry-Mieg
UML2 – Pratique de la modélisation, Pearson, 2010

A. Cockburn
Rédiger des cas d'utilisation efficaces, Eyrolles, 2001

C. Morley, J. Hugues, B. Leblanc
UML pour l'analyse d'un SI, Dunod, 2000

P.A. Muller
Modélisation Objet avec UML, Eyrolles, 1997

P. Roques
UML2 par la pratique, Eyrolles, 6° édition, 2008

P. Roques, F. Vallée
UML2 en action, Eyrolles, 3° édition, 2004

- http://www.omg.org/ (OMG Site)
- www.modelio.org

Objectifs du cours

- Etude des concepts de base du langage UML
- Mise en pratique :
  - TD (5 séances)
  - TP (4 séances) : Etude de cas (Outil Modelio)

Plan

1. Introduction

Contexte

L'informatisation

- Phénomène le plus important de notre époque
- S'immisce partout
- Au coeur de toutes les entreprises
- Système d'information
  - 20% pour le matériel
  - 80% pour le logiciel
- La problématique : essentiellement 'logiciel'

La crise

- Crise de l'industrie du logiciel à la fin des années 1970
  - l'augmentation des coûts
  - les difficultés de maintenance et d'évolution
  - la non fiabilité
  - le non respect des spécifications
  - le non respect des délais
- L'objectif du génie logiciel : optimiser le coût de développement des logiciel (conception & maintenance)

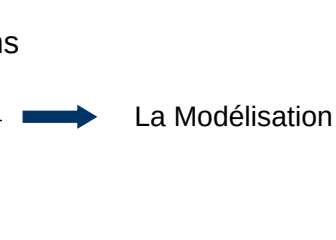
Coût de la maintenance

53% du budget total d'un logiciel est affecté à la maintenance :

- 34% maintenance évolutive (modification des spécifications initiales)
- 10% maintenance adaptative (nouvel environnement, nouveaux utilisateurs)
- 17% maintenance corrective (correction des bugs)
- 16% maintenance perfective (améliorer les performance sans changer les spécifications)
- . . .

Réponse aux problèmes

- Le génie logiciel qui touche au cycle de vie des logiciels :
  - Analyse du besoin
  - Elaboration des spécifications
  - Conceptualisation
  - Développement
  - Phase de tests
  - Maintenance



Modèle = Représentation abstraite et simplifiée d'une entité du monde réel en vue de le décrire, de l'expliquer ou de le prévoir

Avantages de la modélisation

- Mieux comprendre le fonctionnement du système
- Maîtriser la complexité et assurer la cohérence
- Vecteur privilégié pour communiquer
- Mieux répartir les tâches et d'automatiser certaines d'entre elles
- Facteur de réduction des coût et des délais (ex : génération automatique de code)
- Assurer un bon niveau de qualité et une maintenance efficace

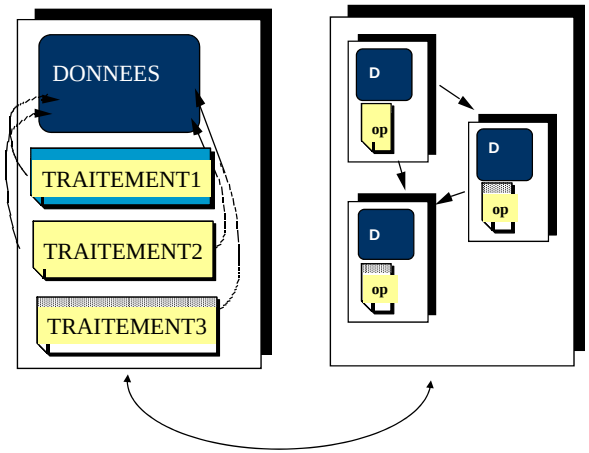
1. Introduction

Conception des SI : deux approches

Conception des SI : deux approches

- **L'approche traditionnelle**
  - Dissociation du problème de la représentation des données et du problème du traitement (l'étude des données et des traitements est effectuée de manière relativement indépendante)
  - Selon le cas : prédominance des données ou des traitements
- **L'approche objet**
  - Accent placé sur les données
  - Organisation sous la forme d'un ensemble d'objets inter-reliés, réunissant en une même unité à la fois la structure des données et des traitements

Approche traditionnelle / objet



Approche traditionnelle : 2 courants de pensée

- **Prédominance des traitements**
  - L'analyse s'intéresse à décrire d'abord les actions (l'activité et les règles d'exécution)
    - Conception fonctionnelle top-down
  - Objectif : identifier les constituants de chaque traitement et la façon dont il va s'exécuter
    - exemples : YOURDON, SADT
- **Prédominance des données**
  - L'analyse s'intéresse à modéliser dans un premier temps les données sous la forme d'entités et de relations.
  - Les opérations susceptibles de consulter ou de modifier l'état des entités et des relations seront étudiées dans un second temps
    - exemples : MERISE, REMORA

En résumé...

- **L'approche traditionnelle** : bien adaptée au développement d'applications peu évolutives
- Ses inconvénients
  - Ses limites sont atteintes lorsque les structures ou les procédures sont partagées
  - une modification de structure doit être reportée sur tous les programmes qui la manipulent
    - ⇒ faible réutilisabilité
    - ⇒ faible extensibilité

Approche objet

- **Accent placé sur les données**
  - identification et structuration des objets, puis identification des actions, puis implémentation
    - exemples : OOA, OOD, OMT, ...
- **Avantages**
  - Possibilité de modéliser les données manipulées et les opérations associées avant de passer à l'implantation
    - ⇒ réutilisabilité
    - ⇒ extensibilité

Intérêt de l'Objet en analyse et conception

- Omniprésence technique de l'Objet
  - dans les langages de programmation, les bases de données, les interfaces graphiques, ... et les méthodes d'analyse et de conception
- Universalité de l'Objet
  - la notion d'objet, plus proche du monde réel, est compréhensible par tous et facilite la communication entre tous les intervenants d'un projet
- Développement de l'orienté objet
  - volonté de réutilisation des composants élémentaires
  - apparition d'un certain nombre de méthodes objets couvrant l'analyse et la conception du cycle du logiciel

Plan

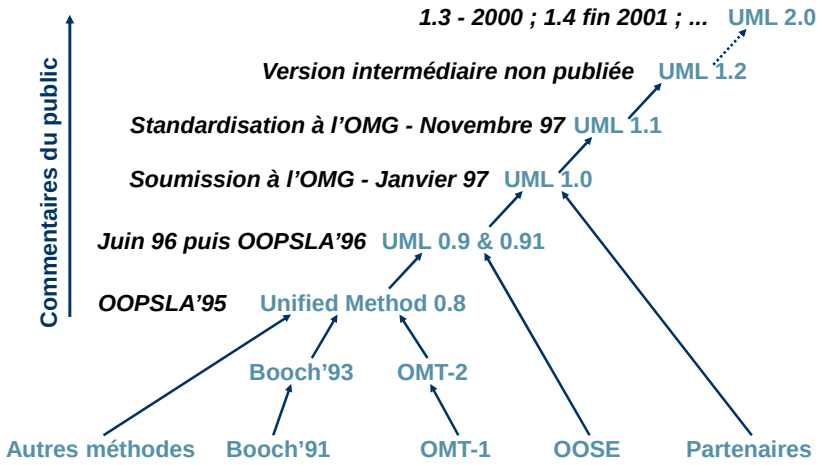
2. UML : historique et notations

Unification des méthodes objet

- Au début des années 90 : une cinquantaine de méthodes objet, liées uniquement par un consensus autour d'idées communes (objets, classes, sous-systèmes, ...)
- Rapprochement des méthodes d'analyse et de conception objet (fusion d'OMT, BOOCH et OOSE)
- Recherche d'un langage commun unique
  - utilisable par toute méthode objet
  - dans toutes les phases du cycle de vie
  - Compatible avec les techniques de réalisation actuelles

  ⇒création d' UML (Unified Modeling Language), choisi par l'OMG comme norme de **langage** de modélisation objet

Historique d'UML



La notation UML

Basée sur les méthodes Booch, OMT et OOSE, elle a été construite autour de l'idée fondamentale de :

- Standardiser les artéfacts de développement
  - modèles, notation et diagrammes
- Ne pas standardiser le processus
  - centré sur l'architecture
  - itératif et incrémental
  - guidé par les *use-cases*
  - ...

Consensus pour la Standardisation

- Objectif : Stabilisation de pratiques industrielles
  - Consensus a maxima
  - Standardisation
- UML ne vise pas l'innovation mais la consensualité

UML

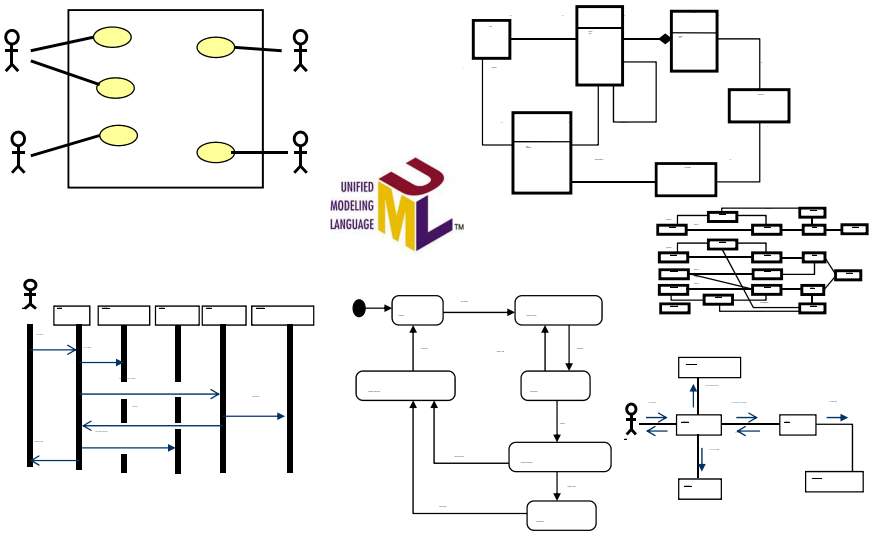
- Modeling

  En pratique l'analyse est toujours orientée par des contraintes de faisabilité qui relèvent de la conception choix du terme de modélisation (modeling)
- Language

  UML n'est pas une méthode, mais un langage

  ==> Une notation
- Des méthodes
  - "The Rational Unified Process"
  - "The Unified Software Development Process"
  - ...
- Des outils
  - Rational Rose, Modelio, Objecteering, Together J, ArgoUML, Poseidon, Papyrus ...

Un ensemble de diagrammes



Les outils UML

- Outils de modélisation
- Génération de code, de documentation,
- De validation, test, …
- Gestion de données
- Transformation de modèles

Plusieurs types de notations

- Notations graphiques
- Notation textuelle

- Signification plus ou moins précise
- Notation standard (mais pas toujours respectée)
- Notation extensible

Remarque

- Notation pas toujours suffisante

Des vues et des diagrammes UML

- Vue statique ou structurelle :
  - Diagrammes de classes, objets, packages
  - Diagramme de structure composite
  - Diagramme de composants et diagramme de déploiement
- Vue dynamique ou comportementale
  - Diagramme des cas d'utilisation
  - Diagramme d'états-transition
  - Diagramme d'activités
  - Diagramme d'interaction : séquence et communication
  - Diagramme global d'interactions
  - Diagramme de temps
- … et des mécanismes
  - Langage de contraintes (OCL)
  - Langage d'actions
  - Langage textuels (notes)
  - …

Vue fonctionnelle

Les différents types de diagrammes UML

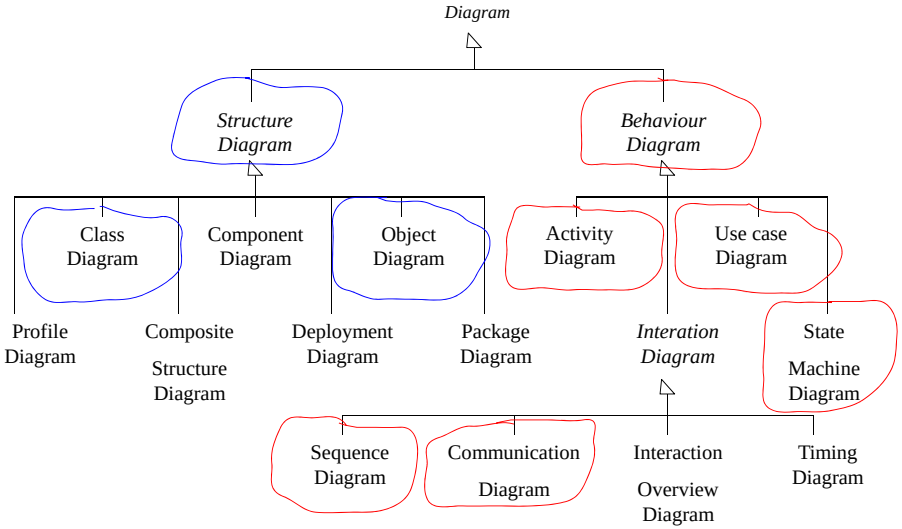


Diagramme de Packages

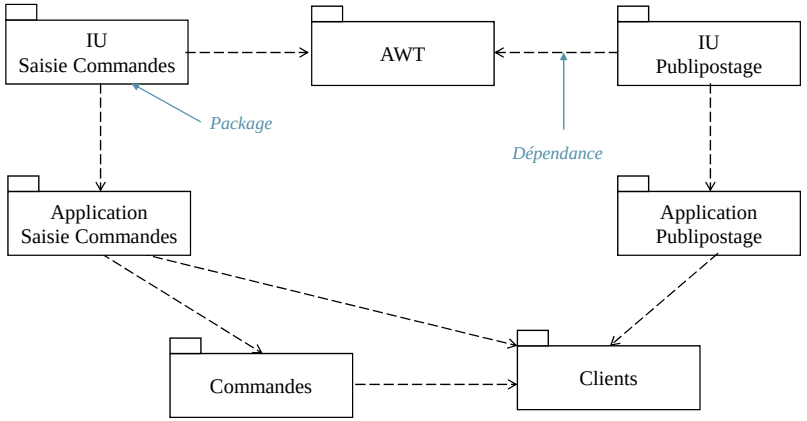


Diagramme de communication

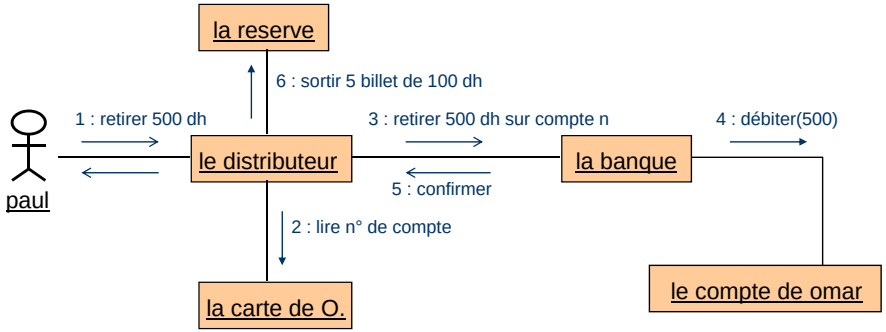


Diagramme global d'interactions

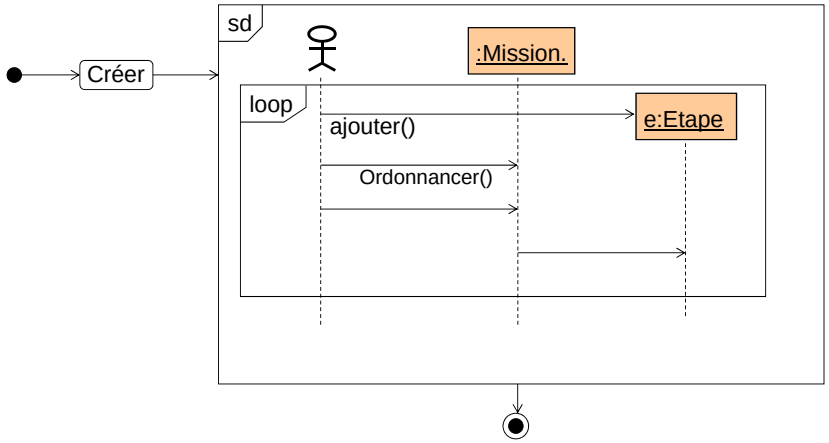
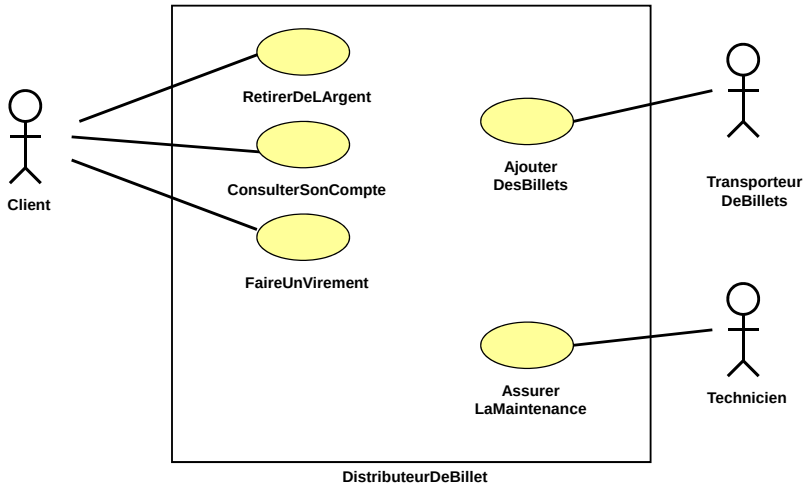
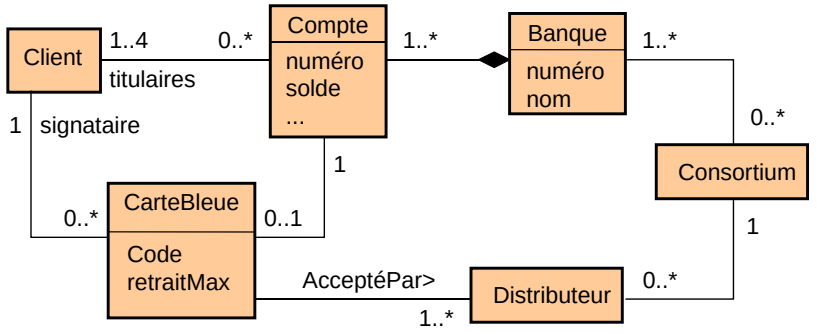


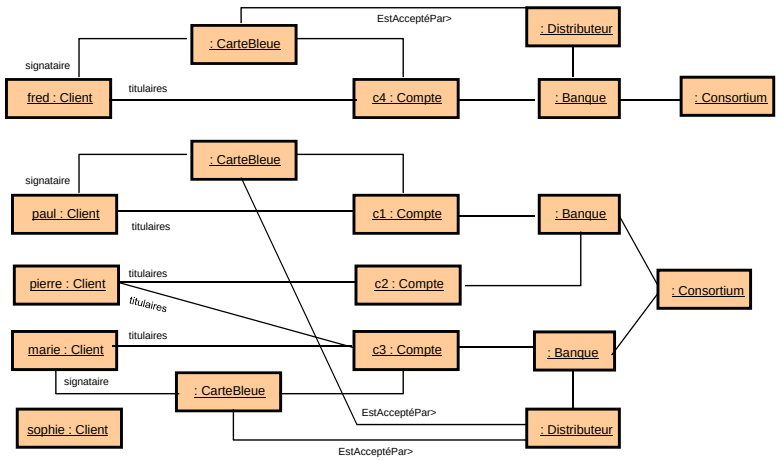
Diagramme des cas d'utilisation



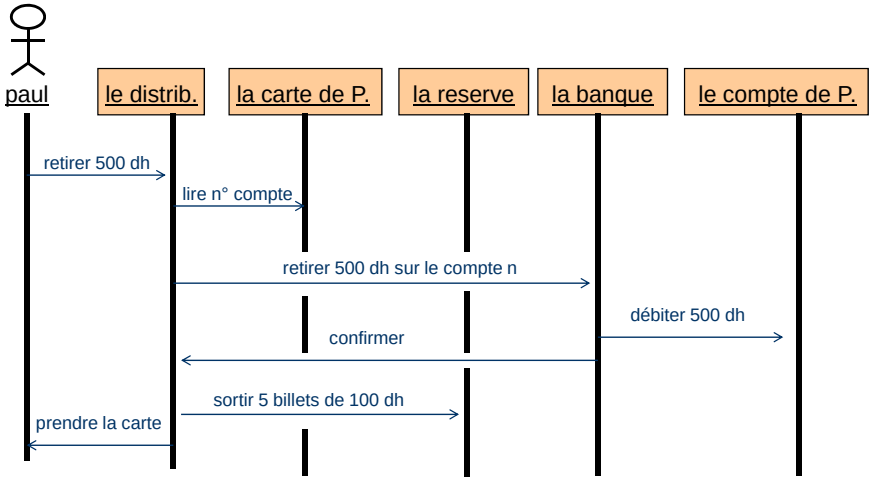
Diagrammes de classes



Diagrammes d'objets



Diagrammes de séquence



Diagrammes d'états

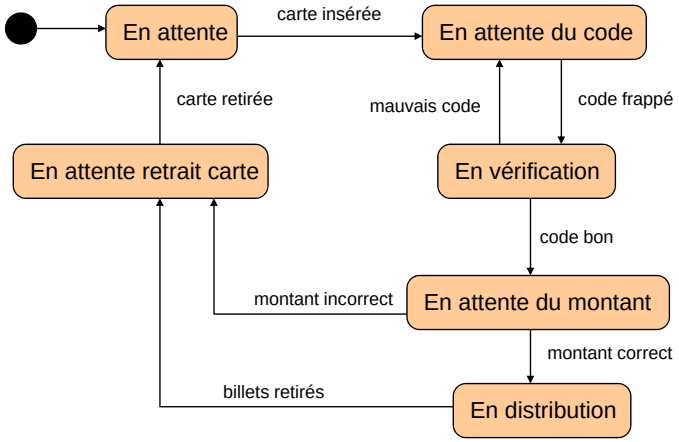
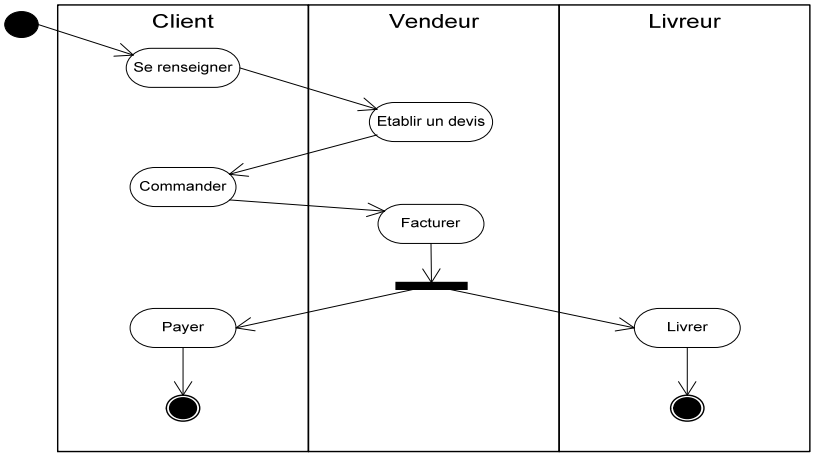
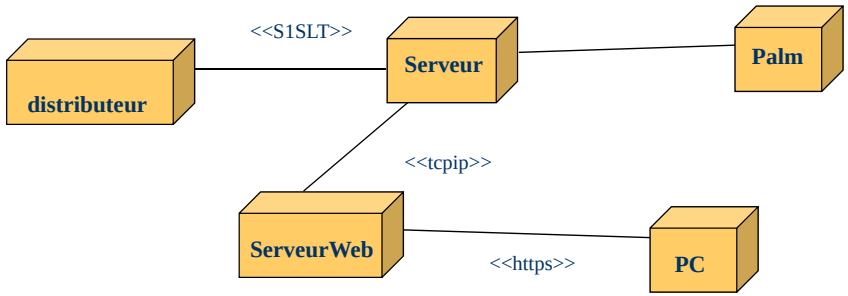


Diagramme d'activités



Diagrammes de déploiement



En résumé

- UML est un langage de modélisation, pas une méthode
- UML est un langage de modélisation objet
- UML convient pour toutes les méthodes objet
- UML est dans le domaine public
- UML est le langage standard de modélisation orientée objet

Plan

1. Introduction
2. UML : Historique et notations
3. **Les Diagrammes UML**
   - **Le diagramme des cas d'utilisation**
   - Le Diagramme de classes
   - Le Diagramme d'interactions
     - Le diagramme de séquence
     - Le diagramme de communication
   - Le diagramme d'états
   - Le diagramme d'activité
   - Le diagramme de composants et diagramme de déploiement
   - Package et communication
4. Mise en œuvre d'UML

3. Les diagrammes UML

Diagramme des use-cases

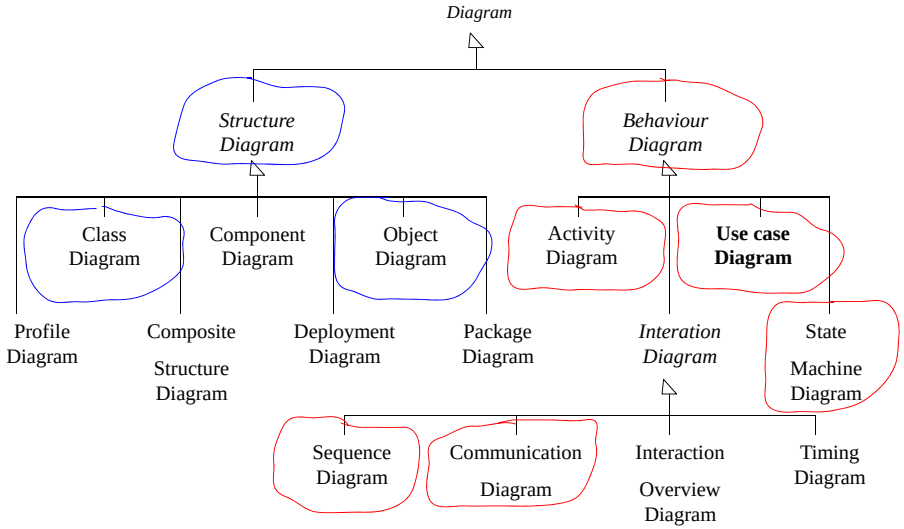
Modélisation / Analyse des besoins

- Le développement ou l'amélioration d'un système doit toujours répondre à un ou plusieurs besoins
- Le travail de modélisation commence par l'identification des besoins
  - Le recueil des besoins implique une bonne compréhension des métiers impliqués
    - Intégration des contraintes et des exigences de chaque métier
- Objectifs de l'analyse des besoins :
  - identifier les frontières du système
  - spécifier les fonctionnalités qu'il doit offrir aux utilisateurs
- L'apport d'UML: diagramme des cas d'utilisation

Modélisation / Analyse des besoins

- Diagramme des cas d'utilisation (use-cases)
  - recenser les grandes fonctionnalités
  - formalisation des besoins
  - représentation graphique des besoins
  - compréhension par tous
  - point de vue utilisateur

Les différents types de diagrammes UML



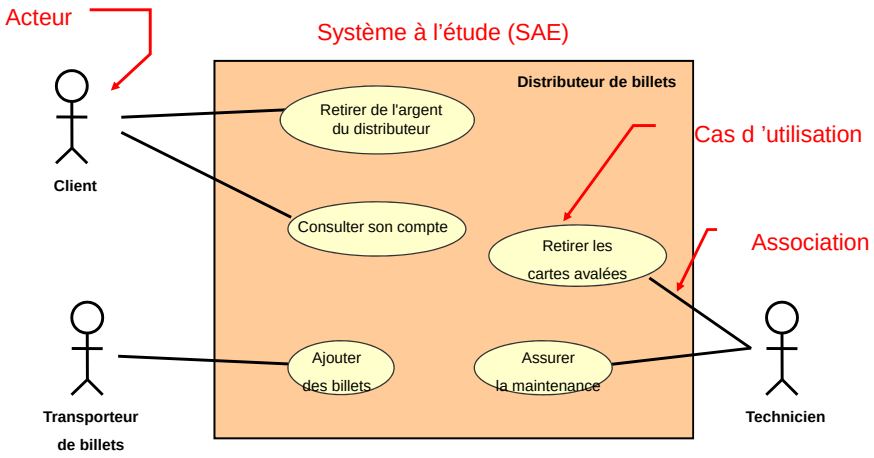
CU / Diagramme de cas d'utilisation (DCU)

- Objectifs des CU
  - Structurer les besoins des **utilisateurs** (modélisation de la vue des utilisateurs)
  - Définir les fonctions du **système** : partition de l'ensemble des besoins fonctionnels d'un système, par catégorie d'utilisateur (cahier des charges)
- Les CU correspondent aux interactions entre le système à concevoir et les acteurs externes
- DCU : notation très simple et compréhensible
  - Basée sur 3 concepts de base :

Acteurs + Cas d'utilisation + Relations

Exemple classique

Diagramme de cas d'utilisation

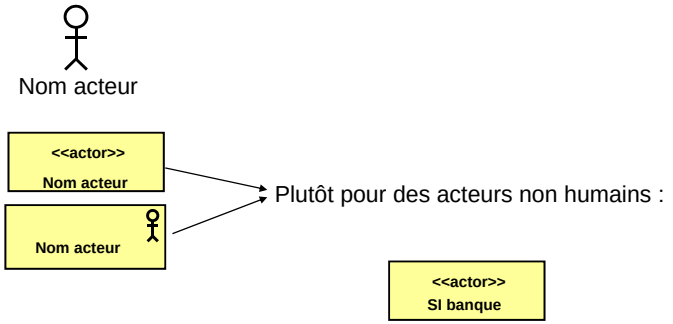


DCU: concepts de base

- **Acteur**
  - Elément externe (au système) qui « interagit » avec le système
  - Rôle qu'un utilisateur joue par rapport au système (client, préparateur commande, opérateur, autre système, ...)
    - Une même personne peut jouer plusieurs rôles
    - Plusieurs personnes peuvent jouer le même rôle
    - Un acteur n'est pas forcément un être humain (ex : distributeur de billets, système informatique,...)
  - Fournit, reçoit, et échange de l'information
    - l'acteur peut consulter ou modifier l'état du système
    - en réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin

DCU : les acteurs - Notation

- **Acteur**



- Rmq : un stéréotype représente une variation d'un élément de modèle existant

Les acteurs d'un cas d'utilisation

- On distingue 2 types d'acteurs : principal / secondaire
- L'acteur principal : celui qui utilise le système
  - demande au système de lui délivrer l'un de ses services,
  - poursuit un objectif par rapport au système ; ex : le client qui vient retirer de l 'argent au GAB
  - L 'acteur principal est en général celui qui déclenche le cas d 'utilisation.
    - Cas particuliers : CU déclenché par un opérateur pour le compte d'une tierce personne ou CU déclenché par un mécanisme temporel
- Les acteurs secondaires :
  - autres participants éventuels (externes au système)
  - fournissent un service au système lors du CU (ex : imprimante, service web, humain informant le système) ou réceptionne une information produite par le système
- Un acteur peut être « acteur principal » dans un CU et « acteur secondaire » dans un autre CU

DCU : les cas d'utilisation

- Définition
  - Les CU ont été proposés par I. Jacobson
  - Les CU sont une technique de description du comportement d'un système selon le point de vue de l'utilisateur
    - Sans imposer le mode de réalisation de ce comportement :

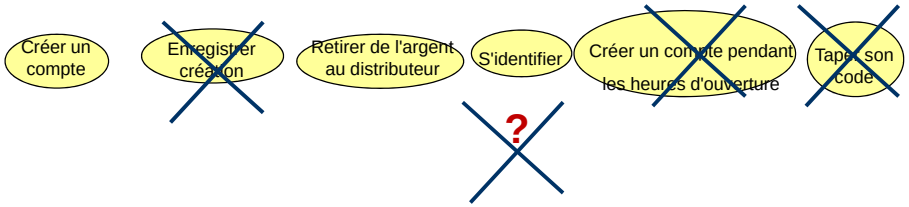
    Description du Quoi ? Sans spécification du Comment ?
  - Un CU correspond à une fonction du système visible par l'acteur
  - Un CU est une séquence d'actions réalisées par le système produisant un résultat observable
    - Regroupe un ensemble de **scénarios** correspondant à un même but

⇒ un CU est une manière spécifique d'utiliser un système

DUC : Cas d'utilisation - Notation

- **Cas d'utilisation**

- **Exemples**



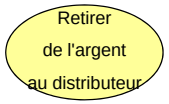
Sous quelle forme se présentent-ils ?

- Fondamentalement les cas d'utilisation se présentent sous une forme textuelle
  - Ils peuvent aussi être élaborés à l 'aide d'organigramme, de diagrammes de séquence ou de diagrammes d'activités, de réseaux de Petri…

☞La forme textuelle présente l'avantage d'être accessible à tous (sans formation spécifique)

Illustration

Cas d'utilisation :



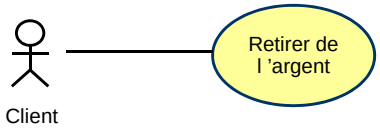
Lorsqu'un *client* a besoin de liquide il peut, en utilisant un distributeur, retirer de l'argent de son compte. Pour cela :

1. Le *client* introduit sa carte dans le lecteur
2. Le *système* demande le code pour l 'identifier
3. Le *client* saisit son code secret. Le système le valide par rapport au code secret crypté lu sur la carte.
4. Le *client* sélectionne Retrait rapide et un montant de retrait.
5. le *système* vérifie qu'il y a suffisamment d 'argent, si c'est le cas, le *système* distribue les billets et débite le compte du client
6. Le *client* prend les billets et retire sa carte

Relations entre éléments de DCU

- 3 types de relations :
  - communication
  - dépendances stéréotypées (inclusion et extension)
  - généralisation/spécialisation

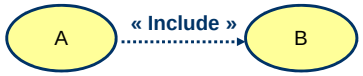
- Relation acteur <-> cas d'utilisation : communication
  - Point de vue **besoin** : représente la possibilité d'atteindre un but
  - Point de vue **système** : représente un échange de messages



Relations entre cas d'utilisation

- Relations cas d'utilisation <-> cas d'utilisation
  - Inclusion : << *include* >>

  Le cas A inclut le cas B (B est une partie *obligatoire* de A)

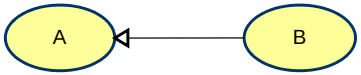


  - Extension : << *extend* >>

  Le cas B étend le cas A (B est une partie *optionnelle* de A)

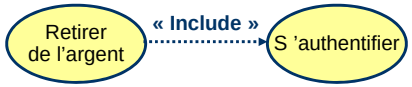
  - Généralisation/spécialisation

  Le cas A est une généralisation du cas B (B est *un cas particulier d*e A)



Relations entre cas d'utilisation

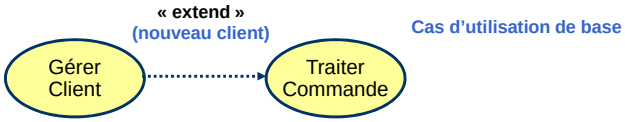
- **La relation d'inclusion** mot-clé « include »
  - Le cas de base en incorpore explicitement un autre, à un endroit spécifié dans ses enchaînements
  - Le cas d'utilisation inclus n'est généralement pas exécuté seul, mais seulement en tant que partie d'un cas de base plus vaste
  - Le cas de base utilise systématiquement les enchaînements provenant du cas inclus
  - Relation qui permet de factoriser des UC correspondants à des fonctionnalités importantes qui servent fréquemment dans différents UC
  - Exemple :



    - Le CU « S'authentifier » est inclus dans le CU « Retirer de l'argent »

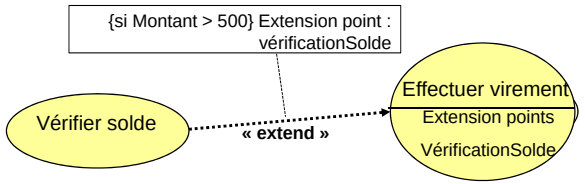
Relations entre cas d'utilisation

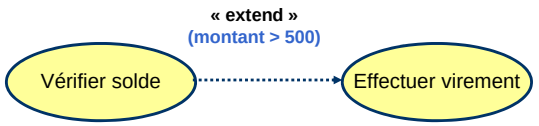
- **La relation d'extension** mot-clé « extend »
  - Le cas de base incorpore implicitement un autre cas, à un endroit spécifié indirectement dans celui qui étend
  - Le cas de base peut fonctionner tout seul, mais il peut également être complété par un autre, sous certaines conditions, et uniquement à certains points particuliers de son flot d'événements appelés **points d'extension**
  - Exemple



    - Le CU « GérerClient» étend le CU « TraiterCommande»

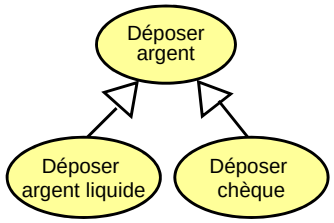
Autre exemple de la relation <<extend>>





Relations entre cas d'utilisation

- **La relation de généralisation / spécialisation**
  - Les CU peuvent être hiérarchisés par généralisation / spécialisation
  - Les CU descendants héritent de la sémantique de leur parent. Ils peuvent comprendre des interactions spécifiques supplémentaires, ou modifier les interactions héritées
  - Exemple :

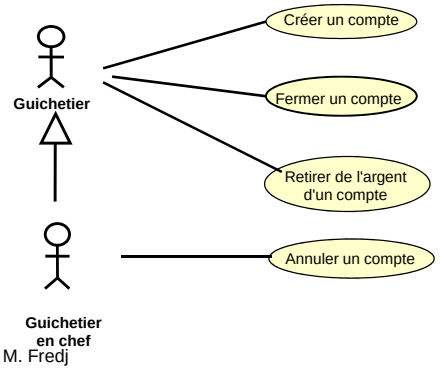


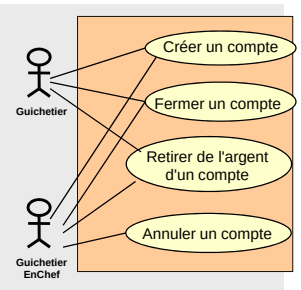
Les CU **« Déposer argent liquide »** et **« Déposer chèque »** sont des spécialisations du CU **« Déposer argent »**

  - La description de cette spécialisation se fera par différence, elle ne contiendra que la partie propre au cas d'utilisation spécialisé.

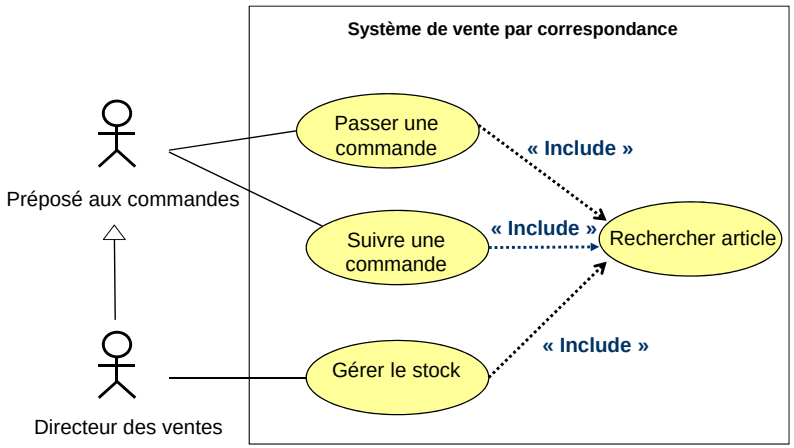
Relations entre acteurs

- Relation acteur <-> acteur
  - Une seule relation la généralisation/spécialisation
  - Les acteurs spécialisés héritent alors des associations de l'acteur ancêtre



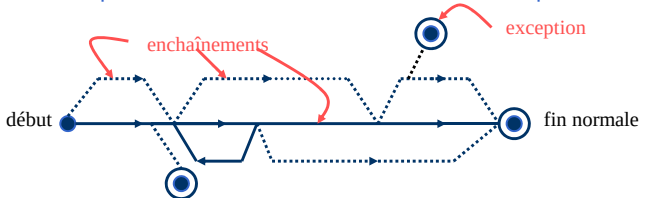


Autre exemple



Cas d'utilisation et scénarios

- **Un scénario :**
  - une manière particulière d'utiliser le système…
    - par un acteur particulier, dans un contexte particulier
  - Il facilite la compréhension des cas et préparent les recettes à venir
    - ☞ scénario = une exécution particulière d'un CU
- Un scénario représente une succession particulière d 'enchaînements, qui s'exécute du début à la fin du cas d'utilisation
- Un cas d'utilisation peut être décrit par plusieurs scénarios
  - Un scénario nominal
  - Eventuellement plusieurs scénarios alternatifs et/ou d'exception



Scénario nominal et scénarios alternatifs

- **Scénario nominal**
  - Scénario de succès typique, même s'il n'est pas forcément le seul chemin menant au succès
  - Scénario rédigé en détail, depuis son déclenchement jusqu'à sa conclusion, et comprenant la réalisation de l'objectif
- **Scénario alternatif (et scénario d'exception)**
  - Tout autre scénario, ou fragment de scénario, rédigé comme une extension au scénario nominal

La fiche de description textuelle d'un cas d'utilisation et de ses scénarios

- Non normalisée par UML
- Structuration couramment préconisée :
  - Un sommaire d'identification (obligatoire)
    - **titre, objectif, résumé, acteur principal et déclencheur**
    - date, version, responsable
  - Une description des enchaînements (obligatoire)
    - Les pré-conditions
    - Les post-conditions
    - Le déroulement normal (scénario nominal)
    - Scénarios alternatifs, d'exceptions **(**variantes possibles et cas d'erreurs)
    - Les interactions entre le système et les acteurs
    - Les informations échangées
    - Les éventuels besoins non fonctionnels

Pré et post conditions

- **Pré-conditions**
  - Définissent ce qui doit être « toujours vrai » avant le début d'un scénario
    - Non testées dans le CU (car tenues pour acquises)
    - Pré supposés importants sur lesquels le rédacteur du CU veut attirer l'attention !
- **Post-conditions**
  - Définissent ce qui doit être « vrai » lorsque le CU se termine avec succès.

Exemple de description textuelle d 'un cas d'utilisation (CU métier)

Titre	**Traiter une commande**
Objectif	Traitement d 'une commande de bout en bout , y compris ses éventuelles modifications.
	Acteurs Principaux : le client, le préparateur du dépôt Evt déclencheur : la demande client
Descriptif	Deux variantes : - commande directe, saisie par le client lui-même - commande indirecte, saisie par le préparateur 1) 2)
	Règles : - on ne fait pas de livraisons partielles - une commande est facturée dès livraison
	Documents : ---> commande <--- facture

Autre exemple : description du CU système « Enregistrer un emprunt »

- **Titre : Enregistrer un emprunt**
- **Objectif** : Enregistrer la sortie d'un ouvrage et associer l'emprunt à son emprunteur
- **Acteur principal** : Employé bibliothèque
- **Evt déclencheur** : Demande d'emprunt réceptionnée à la banque d'accueil
- **Acteurs secondaires** : néant
- **Résumé** : Après avoir vérifié si le demandeur est bien adhérent de la bibliothèque et s'il a la possibilité d'emprunter, l'employé recherche, pour chaque ouvrage demandé, si un exemplaire de l'ouvrage est disponible puis enregistre l'emprunt.
- **Pré condition** : Le système de gestion des emprunts est en état de fonctionnement.
- **Post-condition** (scénario nominal) : la fiche de prêt de l'adhérent est à jour, les exemplaires d'ouvrages empruntés sont notés comme « En cours de prêt ».

Identifier les cas d'utilisation

- **Attention !**
  - Un cas d'utilisation n'est ni une fonction ni une transaction
    - Une erreur fréquente consiste à vouloir descendre trop bas en termes de granularité
    - Un cas d'utilisation représente un ensemble de séquences d'actions réalisées par le système,
    - Le cas d'utilisation ne doit pas se réduire à une seule séquence, et encore moins à une seule action !
- ☞ **Un système sera rarement décrit par plus d'une vingtaine de cas d'utilisation**

Problèmes de la granularité

- A quel niveau décrire les cas d'utilisation ?
- Bonne question ... mais pas de réponse
- Trop haut
  - trop loin du système
  - trop abstrait et "flou"
  - trop complexe à décrire
- Trop bas
  - trop de cas d'utilisation
  - trop près de l'interface
  - trop loin des besoins métiers
- Conclusion : choisir le "bon" niveau ...

La démarche : Processus Unifié

1. Définir le modèle de cas d'utilisation
   1. Trouver les acteurs
   2. Décrire brièvement chaque acteur
   3. Trouver les cas d'utilisation
   4. Décrire brièvement chaque cas d'utilisation
   5. Décrire le modèle comme un tout
2. Définir des priorités et les risques entre CU
3. Détailler chaque CU (en tenant compte des priorités et des risques)

Exemple description détaillée d'un CU

Retirer de l'argent du distributeur

Précondition :
Le distributeur contient des billets, il est en attente d 'une opération, il n'est ni en panne, ni en maintenance

Début : lorsqu'un client introduit sa carte bancaire dans le distributeur.

Fin : lorsque la carte bancaire et les billets sont sortis.

Postcondition :
Si de l 'argent a pu être retiré, la somme d'argent sur le compte est égale à la somme d'argent qu'il y avait avant, moins le montant du retrait. Sinon la somme d'argent sur le compte est la même qu'avant.

Exemple description détaillée d'un CU

Retirer de l'argent du distributeur

Scénario nominal :
(1) le *client* introduit sa carte bancaire
(2) le *système* lit la carte et vérifie si la carte est valide
(3) le *système* demande au client de taper son code
(4) le *client* tape son code confidentiel
(5) le *système* vérifie que le code correspond à la carte
(6) le *client* choisit une opération de retrait
(7) le *système* demande le montant à retirer
…
Variantes :
(A) *Carte invalide* : au cours de l'étape (2) si la carte est jugée invalide, le système affiche un message d'erreur, rejette la carte et le cas d'utilisation se termine.
(B) *Code erroné* : au cours de l'étape (5) ...

Exemple description détaillée d'un CU

Retirer de l'argent du distributeur

Contraintes non fonctionnelles :

(A) *Performance* : le système doit réagir dans un délai inférieur à 4 secondes, quelque soit l'action de l'utilisateur.

(B) *Résistance aux pannes* : si une coupure de courant ou une autre défaillance survient au cours du cas d'utilisation, la transaction sera annulée, l'argent ne sera pas distribué. Le système doit pouvoir redémarrer automatiquement dans un état cohérent et sans intervention humaine.

(C) *Résistance à la charge* : le système doit pouvoir gérer plus de 1000 retraits d'argent simultanément
...

Autres formes de description d'un cas d'utilisation et de ses scénarios

- Les principaux scénarios d'un cas d'utilisation peuvent aussi être décrits à l'aide de :
  - diagrammes de séquence (pour un scénario) ou
  - diagrammes d'activités (pour représenter la dynamique d'un CU avec plusieurs scénarios)

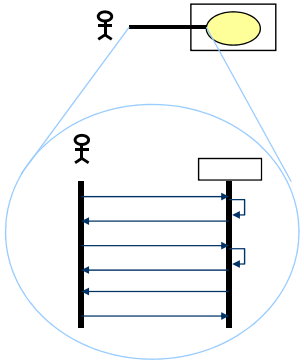
☞ **CU « boîte noire »**
représentation des interactions entre l'acteur principal, le système et les éventuels acteurs secondaires
Diagramme de Séquence Système (DSS)

☞ **CU « boîte blanche »**
représentation des interactions entre les objets ou entre les acteurs internes au système à l'étude
Diagramme de Séquence Détaillé

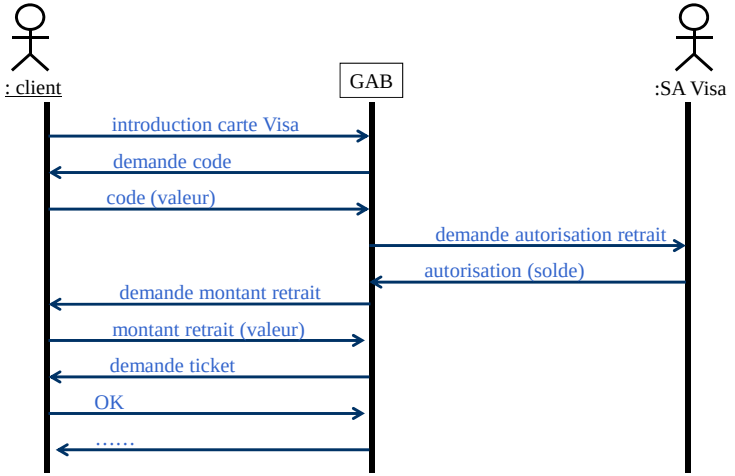
Diagrammes de séquence système (DSS)

- Diagramme de séquence :
  - L'une des notations UML, une notation générale
  - Peut être utilisé dans de nombreux contextes
  - Permet de décrire une séquence des messages échangés entre différents objets
  - Différents niveaux de détails
- Pour décrire un scénario simple, deux objets : l'acteur et le système

  – Le système informatique est considéré comme une « boîte noire »

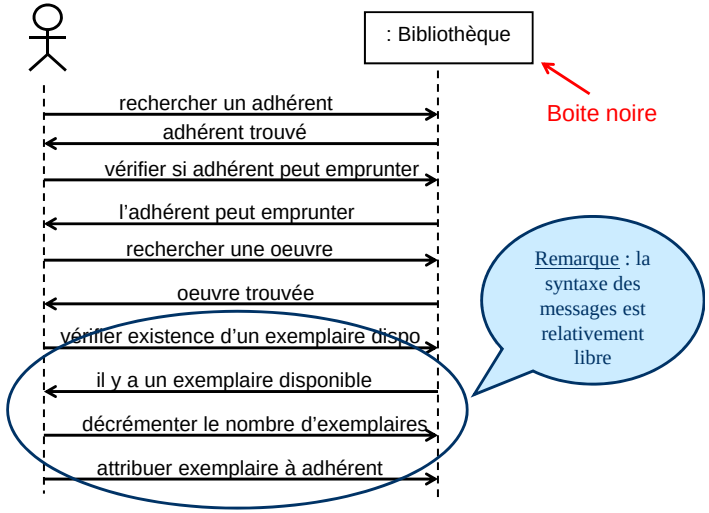


Exemple DSS : Retirer de l'argent d'un GAB



Scénario nominal : Retirer de l'argent dans un GAB avec une carte Visa

Autre exemple : DSS de l'emprunt d'un livre



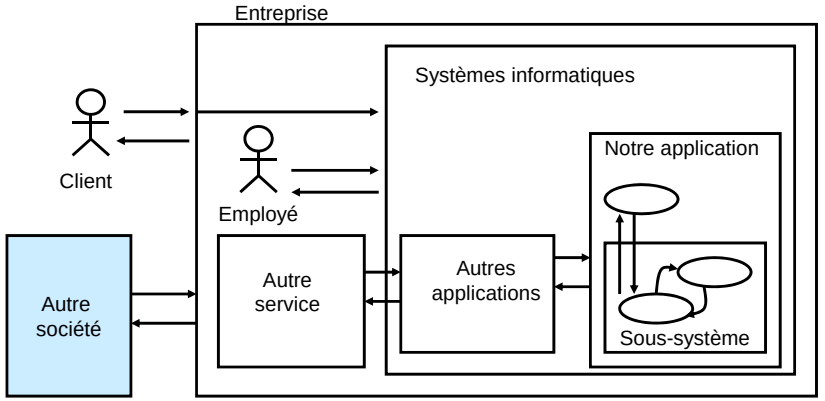
Mise en oeuvre des cas d 'utilisation...

- Trois concepts fondamentaux à maîtriser :
  - La portée de conception
    - Quel est le système à l 'étude ?
  - L'acteur principal
    - Qui détient l 'objectif ?
  - Le niveau d 'objectif
    - A quel niveau (faible ou élevé) se situe cet objectif ?

La portée d'un cas d'utilisation

- **La portée d'un cas d 'utilisation désigne l'étendue du système à l'étude (SAE) :**
  - L'entreprise (ou éventuellement un service de l 'entreprise)
    - comportement de l 'entreprise dans son ensemble lors de la réalisation de l 'objectif de l 'acteur principal.
      (ex : l 'entreprise MonEntreprise qui traite les demandes des clients de la prise de commande jusqu'à la livraison)
  - Le système informatique (ou un ensemble de systèmes info)
    - partie de matériel ou de logiciel faisant l 'objet de la conception.
      (ex : le système Gestcom servant à enregistrer les commandes client)
  - Un sous-système
    - partie du système principal
      (ex : le sous-système de recherche d'un client à partir de son nom)

La portée de conception peut être de n'importe quelle taille



- Un CU « métier » a pour portée l'entreprise.
- Un CU « système » a pour portée le système informatique
- Un CU « composant » a pour portée un composant ou un sous-système du système en cours de conception.

Illustration : le cas du GAB (1/2)

- Quel est le système à l'étude (SAE) ?
  - Le GAB
  - La banque
  - …
- Selon la portée de conception retenue (ex : GAB ou banque), les acteurs suivants sont-ils (ou non) des acteurs du système ?

- GAB
- Client
- Carte de retrait
- Banque
- Panneau frontal du GAB
- Réparateur
- Imprimante
- SI banque
- Guichetier
- Braqueur
- ….

Illustration : le cas du GAB (2/2)

Acteurs	SAE = GAB	SAE = banque
GAB		
Client		
Carte de retrait		
Banque		
Panneau frontal		
Propriétaire banque		
Réparateur		
Imprimante		
SI banque		
Guichetier		
Braqueur		

Niveaux d'objectifs et intérêt des cas d'utilisation

- Les CU de niveau « stratégie » servent à situer les cas d 'utilisation de niveau inférieur dans un contexte plus large
  - Ils sont en nombre restreint et résument les intérêts de trois ou quatre acteurs principaux fondamentaux (le client vis à vis de l 'entreprise, le service marketing vis à vis du S.I., …)
- Les CU de niveau « objectif utilisateur » sont importants à décrire
  - car ils correspondent aux fonctionnalités métier (exigences fonctionnelles du système à construire)
- Les CU de niveau « sous-fonction » ne doivent pas être décrits systématiquement...
  - Ils décrivent les sous-étapes nécessaires pour la réalisation d'un objectif utilisateur
  - uniquement s'ils permettent une meilleure lisibilité ou si un grand nombre de cas d 'utilisation « objectif utilisateur » en font usage,
  - sinon les intégrer dans un autre cas d 'utilisation.

Exemple de CU métier (portée entreprise) de niveau stratégique (A. Cockburn)

- **Titre : Traiter une demande d'indemnisation**
- ***Portée (SAE) :*** Compagnie d'assurance
- ***Acteur Principal :*** Assuré
- ***Déclencheur :*** Un sinistre a été déclaré auprès de la compagnie d'assurance
- ***Scénario nominal :***
  - Un assuré déclare un sinistre auprès de la compagnie d'assurances.
  - L'employé reçoit et affecte la déclaration à un régleur de sinistres.
  - Le régleur de sinistre désigné :
    - Mène une enquête
    - Évalue les dommages
    - Fixe le montant des prévisions
    - Négocie la demande d'indemnisation
    - Liquide la demande d'indemnisation et clôt le dossier
  - ***Post condition (en cas de succès) :*** la demande est liquidée et le dossier est clos

Exemple de CU système de niveau objectif utilisateur (A. Cockburn)

- **Titre : Enregistrer un sinistre**
- ***Portée (SAE) :*** Le système informatique de saisie des sinistres
- ***Acteur Principal :*** L'employé
- ***Déclencheur :*** L'employé commence la saisie d'un sinistre
- ***Pré conditions*** : L'employé a ouvert une session
- ***Scénario nominal :***
  - L'employé saisit le numéro de police de l'assuré ou son nom et la date de l'incident. Le système intègre les informations disponibles sur la police et indique que la demande d'indemnisation est reliée à la police.
  - L'employé saisit les informations élémentaires sur le sinistre. Le système attribue un numéro de demande d'indemnisation.
  - …
  - L'employé sélectionne et désigne un régleur de sinistre
  - L'employé confirme qu'il a terminé. Le système enregistre et génère une notification à l'agent.
  - ***Post condition (en cas de succès) :*** Les informations concernant le sinistre sont saisies et stockées

Le regroupement des cas d'utilisation en package (ou paquetage)

- **Dans le cadre d'un projet, on peut regrouper les cas d'utilisation suivant différentes stratégies :**
  - par domaine d'expertise métier (intuitif)
  - par acteur (simple si chaque cas d'utilisation est relié à un acteur)
  - par lot de livraison (si développement incrémental)
- **Le mécanisme générique de regroupement d'éléments en UML s'appelle le « package » (ou paquetage)**

  → correspond à une notion de sous-modèle

Qu'est ce qu'un package ?

- Un package UML représente un espace de nommage qui peut contenir :
  - des éléments de modèle
  - des diagrammes qui représentent les éléments du modèle
  - d 'autre packages
- Les éléments contenus dans un package :
  - doivent présenter un ensemble fortement cohérent
  - sont généralement de même nature et de même niveau sémantique

Exemple de regroupement

- Pour un système de suivi de commande (de la prise de commande à sa livraison et facturation) on pourra utiliser un critère de regroupement selon le domaine d'expertise métier :

Cas d 'utilisation	Acteurs principaux	Package
Gérer commande	Réceptionniste	**Gestion clientèle**
Gérer client	Réceptionniste	
Consulter en-cours	Client Réceptionniste	
Planifier tournées	Répartiteur Chauffeur	**Gestion tournée**
Suivre tournée	Répartiteur Chauffeur	
Facturer	Progiciel comptabilité Comptable	**Comptabilité**
Suivre règlements	Comptable	

Etc….

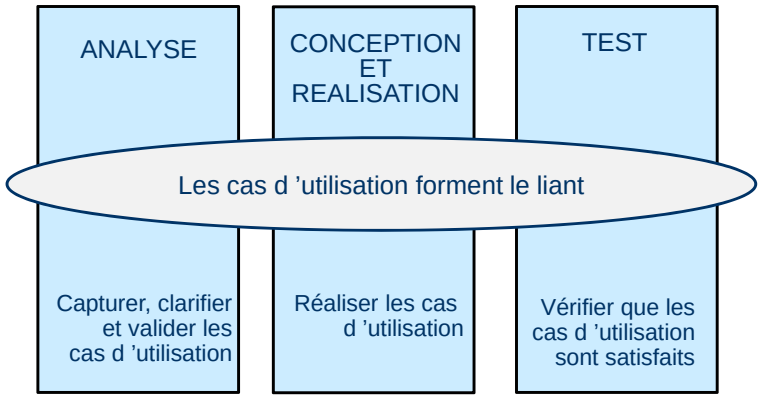
Le modèle de cas d'utilisation

- **Un modèle de cas d'utilisation peut contenir :**
  - plusieurs diagrammes de cas d'utilisation
  - plusieurs descriptions textuelles
- **Il permet d'avoir une vision globale du système en adoptant un point de vue externe :**
  - les fonctions essentielles du système
  - les limites du système
  - le système dans son environnement
- Il permet éventuellement de définir des priorités entre cas d 'utilisation

Le modèle de cas d'utilisation

- **La description d'un cas d 'utilisation peut être plus ou moins détaillée selon le contexte d'utilisation**
  - Objectif du cas d'utilisation (orienter la discussion sur le futur système logiciel, décrire les exigences fonctionnelles, documenter la conception du système)
  - Taille et dispersion des équipes
- Commencer la description détaillée des cas d'utilisation par les plus prioritaires !
- **Les cas d 'utilisation ne doivent pas se chevaucher**

Les cas d'utilisation sont au cœur du processus de développement...



(P.A. Muller - Modélisation objet avec UML)

Recommandations

- Un seul acteur principal par cas d'utilisation.
  - acteur *principal :* celui pour qui le cas d'utilisation produit un résultat observable. Par opposition, nous qualifions d'acteurs *secondaires* les autres participants du CU
- Dans la mesure du possible, disposez les acteurs principaux à gauche des cas d'utilisation, et les acteurs secondaires à droite
- Utilisez la forme graphique du *stick man* pour les acteurs humains, et la représentation rectangulaire avec le mot-clé <<actor>> pour les systèmes connectés
- Éliminez les acteurs «physiques» au profit des acteurs «logiques» :
  - l'acteur est celui qui bénéficie de l'utilisation du système

Recommandations (suite)

- Répertoriez en tant qu'acteurs uniquement les entités externes
  - (et pas des composants internes au système étudié) qui interagissent directement avec le système
- Ne confondez pas rôle et entité concrète.
  - Une même entité externe concrète peut jouer successivement différents rôles par rapport au système étudié, et être modélisée par plusieurs acteurs. Réciproquement, le même rôle peut être tenu par plusieurs entités externes concrètes, qui seront alors modélisées par le même acteur.
- Utilisez la relation d'inclusion entre cas d'utilisation pour éviter de devoir décrire plusieurs fois le même enchaînement, en factorisant ce comportement commun dans un cas d'utilisation supplémentaire inclus.

Recommandations (suite)

- N'abusez pas des relations entre CU (surtout extension et généralisation) :
  - elles peuvent rendre les DCU difficiles à décrypter pour les experts métier qui sont censés les valider.
- Limitez à 20 le nombre de vos cas d'utilisation de base (en dehors des cas inclus, spécialisés, ou des extensions)
  - Avec cette limite arbitraire, on reste synthétique et on ne tombe pas dans le piège de la granularité trop fine des cas d'utilisation.
  - Un cas d'utilisation ne doit donc pas se réduire à une seule séquence, et encore moins à une simple action.