UML
Unified Modeling Language

2ème partie

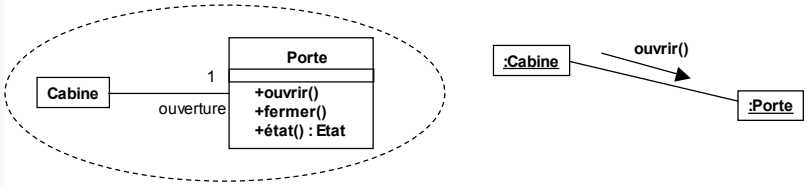
M1 MIAGE - SIMA - 2005-2006
Yannick Prié
UFR Informatique - Université Claude Bernard Lyon 1

Plan du cours

- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- **Diagrammes d'interaction**
- Diagrammes d'activité
- Diagrammes de machines d'état
- Diagrammes de composants et de déploiement
- Autres diagrammes UML
- Autres diagrammes non UML
- Autres points liés à UML

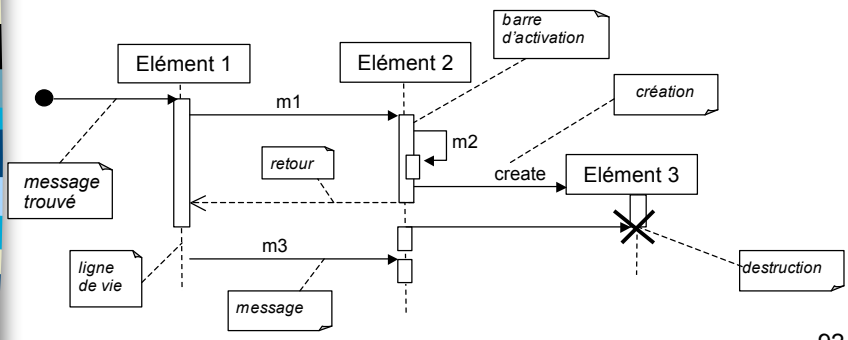
Collaborations et interactions

- Collaboration
  - ensemble de rôles joués par des classes, contexte d'interaction
- Interaction
  - communication entre instances des éléments d'une collaboration
  - ensemble partiellement ordonné de messages
  - plusieurs interactions possibles pour une même collaboration
- Éléments d'une interaction
  - participants (UML1 : objets, UML2 : souvent objets)
  - liens (supports de messages)
  - messages (déclenchant des opérations)
  - rôles joués par les extrémités de liens



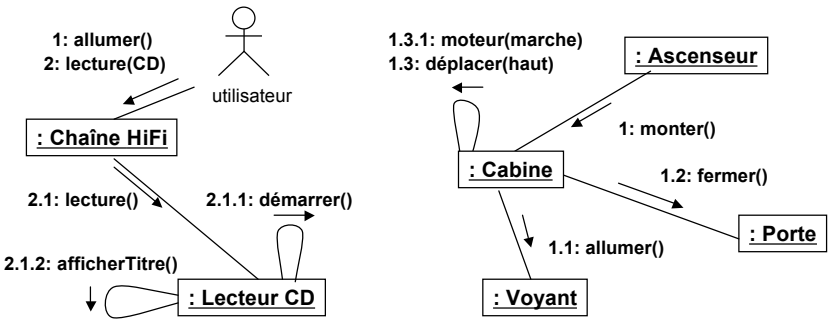
Diagrammes de séquences

- Interactions entre éléments dans une séquence *temporelle*
  - aspect chronologique ne rendant pas compte explicitement du contexte
  - permet de bien montrer qui fait quoi dans une interaction
- Description de scénarios typiques et des exceptions



Diagrammes de communication
(UML1 : diagrammes de collaboration)

- Diagramme d'objets rendant compte de la dynamique
  - structure spatiale permet la collaboration d'objets
  - dimension temporelle : ordre des messages
    - numérotation pointée



Petit exercice à faire en classe

- Dessiner un diagramme de communication impliquant le passage de la balle entre deux tortues d'équipes différentes.

Utilisation

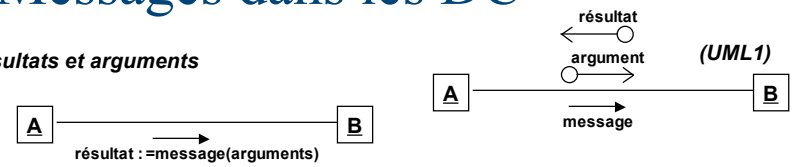
- Etudier/spécifier le comportement
  - du système au sein d'un cas d'utilisation
    - se concentrer sur les événements du système considéré comme boîte noire
      - diagramme de séquence système (exemple plus loin)
  - de plusieurs objets au sein d'un cas d'utilisation
    - réalisations de CU comme des interactions dans une société d'objets
      - Conseil : dessiner diagrammes de classes et d'interaction en même temps
- Illustrer/étudier un fonctionnement
  - diagramme qui traverse les couches : de l'IHM aux données
  - rétro-ingénierie

Messages

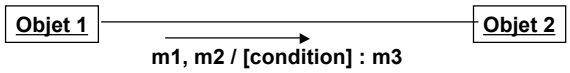
- Matérialisation d'une communication avec transmission d'information entre
  - émetteur (source)
  - récepteur (destination)
- Un message déclenche
  - une opération,
  - l'émission d'un signal
  - la création/destruction d'un objet
- Deux types principaux
  - appel de procédure ou flot de contrôle emboîté (retour implicite)
    - déplacer()
  - flot de contrôle asynchrone
    - démarrer()
  - autres : à plat, dérobant (réception si attente), minuté (message actif pendant Dt)

Messages dans les DC

Résultats et arguments

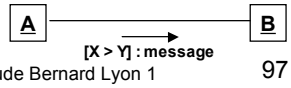


Synchronisation



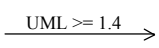
Itérateurs

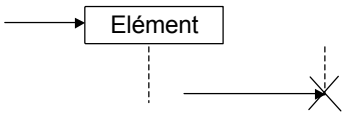
Garde



Messages dans les diagrammes de séquences

- Notation résultat = message(arguments)
- Echange de messages
  - flèches d'appel standard
    - blocage de l'émetteur en attendant la réponse
  - flèche d'appel asynchrone
    - pas d'attente du retour, poursuite de la tâche
  - Retour
  - Message de création
  - Message de destruction

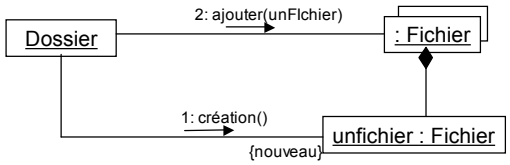




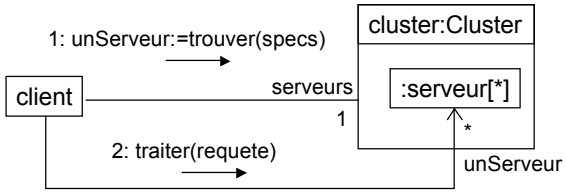
- Lancement de l'interaction venant de l'extérieur
  - 1er message = « message trouvé »

Gestion de collections

- Créer et ajouter (UML1 : multi-objet)



- Récupérer et utiliser (UML2 : structure composite)

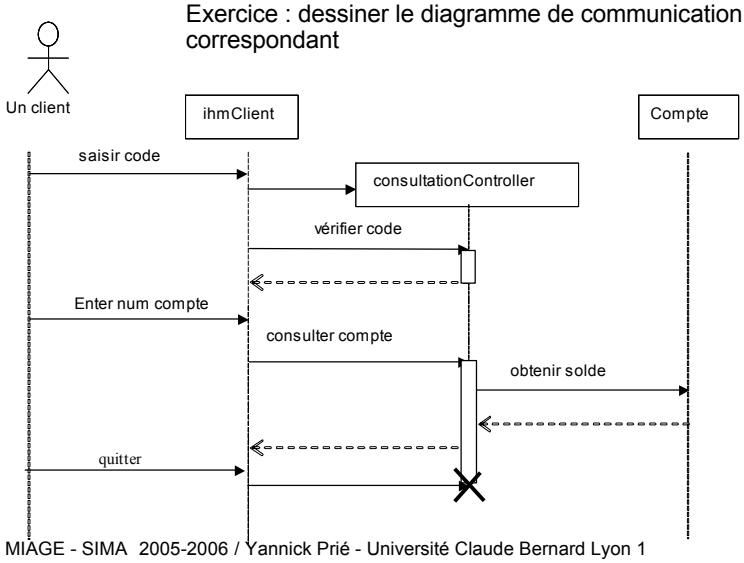


Raffinements DS/UML2

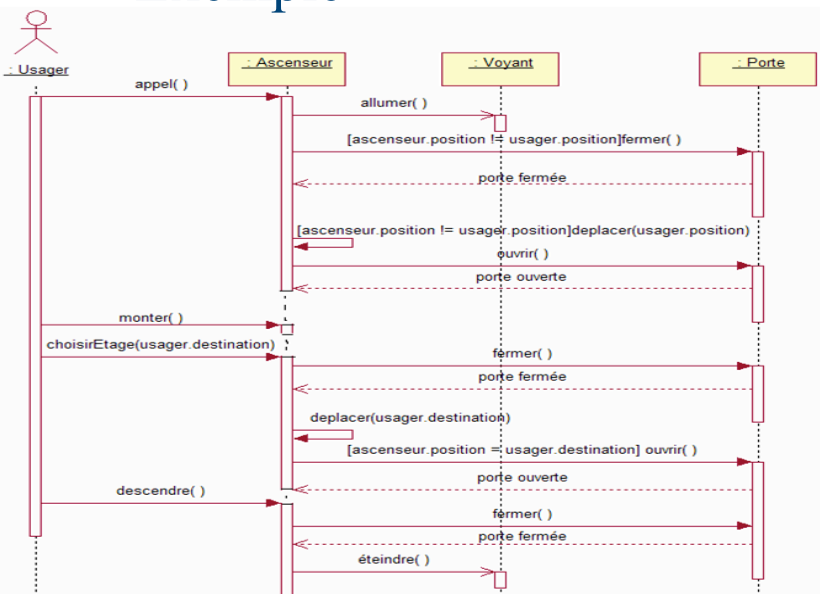
- Diverses possibilités de participants
  - interfaces : spécifier quelle interface participe à l'interaction
  - classes : pour appeler une méthode de classe
- Représentation polymorphisme / classe abstraite

Equivalence entre diagrammes

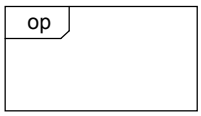
Exercice : dessiner le diagramme de communication correspondant



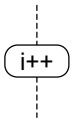
Exemple



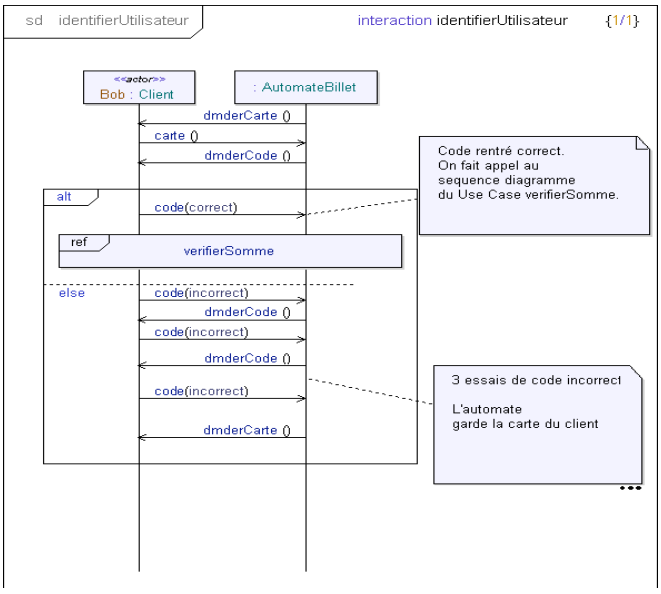
Cadre d'interaction



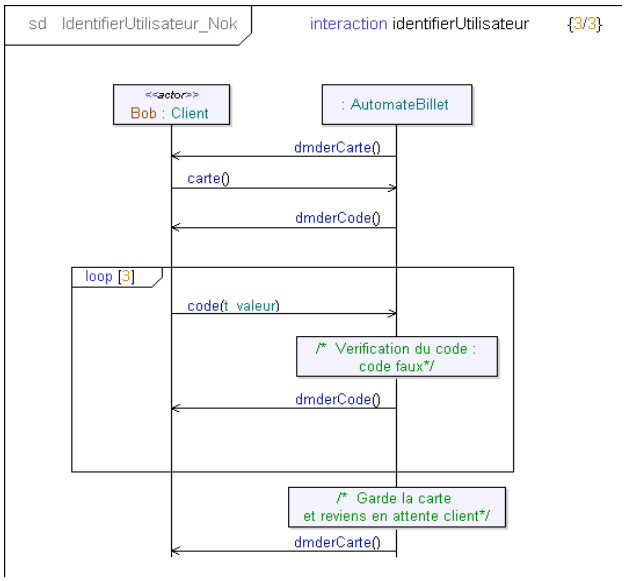
- Cadre nommé par un opérateur qui entoure un fragment critique du DS
  - alt
    - fragment alternatif, conditions dans les gardes
  - loop
    - fragment à répéter tant que la condition de garde est vrai
    - notion de boîte d'action avec itérateur
  - opt
    - fragment optionnel exécuté si la garde est vraie
  - par
    - fragments qui s'exécutent en parallèle
  - region
    - region critique dans laquelle un seul thread doit s'exécuter
  - ref
    - passage à un autre diagramme de séquence
- Attention
  - ne pas représenter des algorithmes : trop compliqué



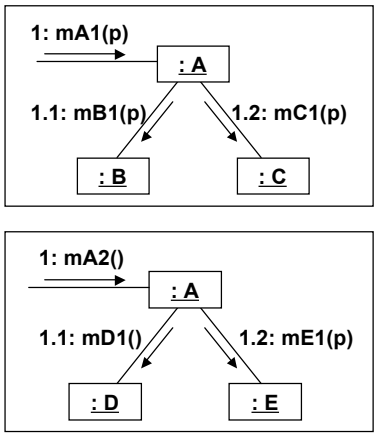
alt



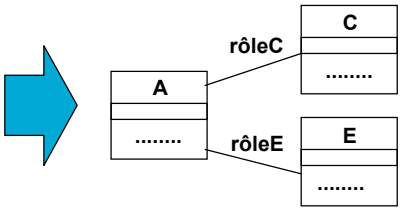
loop



Déduire structure et responsabilité des diagrammes d'interaction



- Liens
  - ➜ associations
- Messages
  - ➜ opérations

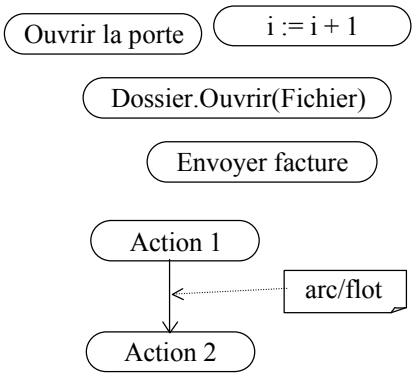


Plan du cours

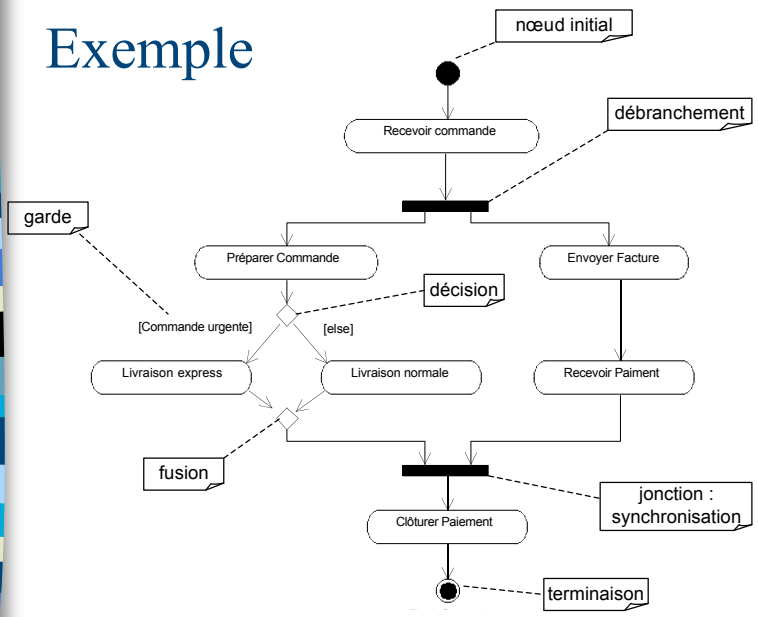
- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- Diagrammes d'interaction
- **Diagrammes d'activité**
- Diagrammes de machines d'état
- Diagrammes de composants et de déploiement
- Autres diagrammes UML
- Autres diagrammes non UML
- Autres points liés à UML

Diagrammes d'activité

- Diagramme d'activité
  - présenter les activités séquentielles d'un processus
  - activité = suite d'action
- Action
  - travail à réaliser
  - nœud du graphe
- Transition
  - contrainte d'enchaînement
  - relation du graphe
- Raffinements
  - débranchements / jointures
  - décisions / fusions
  - entrée / terminaison
  - ressources utilisées (objets)



Exemple



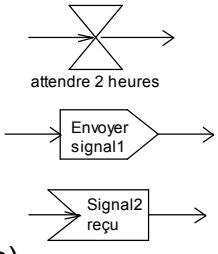
Petit exercice à faire en classe

- Modéliser les activités autour d'un enseignement de l'UFR informatique.

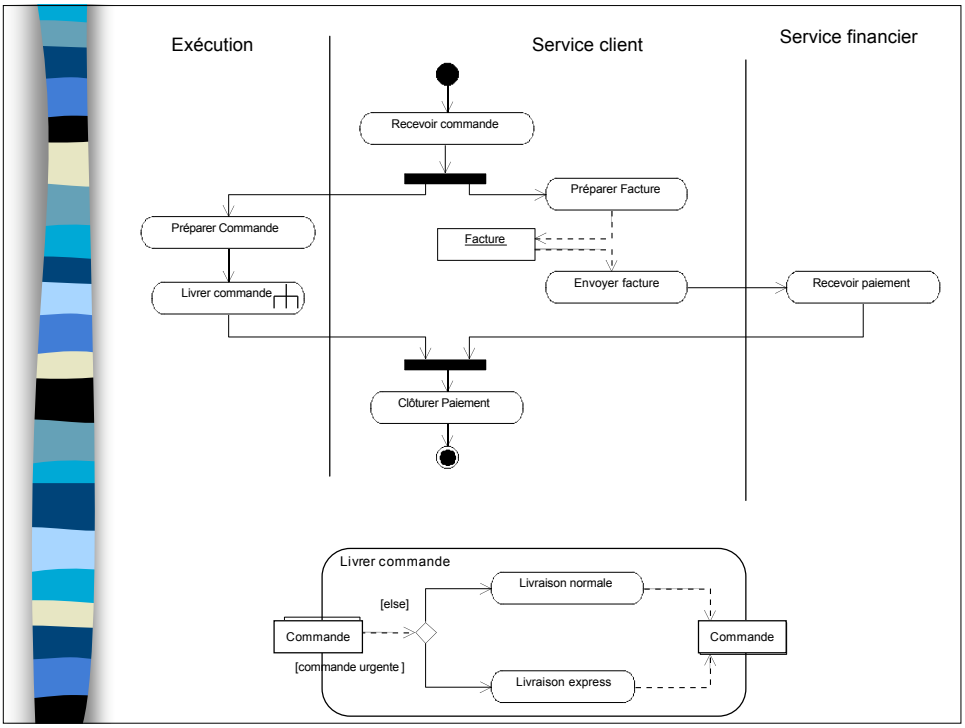
Utilisation pour modéliser

- Les processus métier de l'organisation
  - qui fait quoi, où
  - les enchaînement d'activité (workflow)
- Les flots de données
  - DFD (Data Flow Diagram) en UML
- La logique procedurale
  - algorithmes complexes, parallèles
  - organisation séquentielle globale des activités de plusieurs objets
    - vs. diag. machines d'états : un objet

Diagrammes avancés

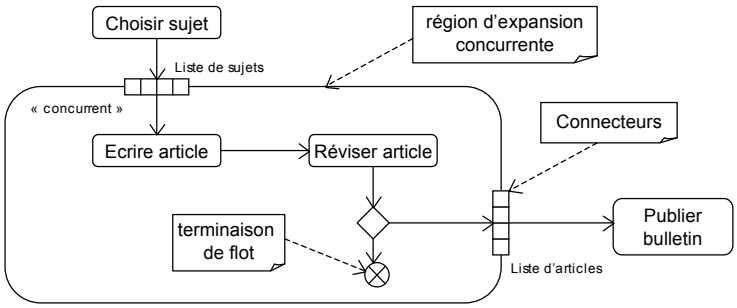


- Actions liées à des signaux
  - délai
  - envoi / réception
- Utilisation d'objets
  - en entrée ou sortie d'action
- Partitions (UML1 : *swimlanes*, travées)
  - montrer les responsabilités au sein du mécanisme ou d'une organisation
- Décomposition des actions
  - appeler une sous-activité (un autre diagramme d'activité) dans une action



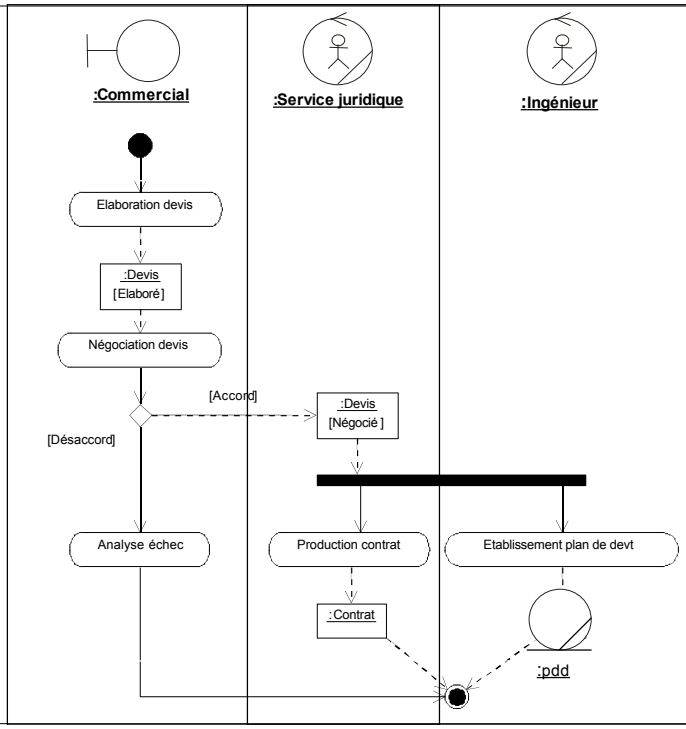
Connecteurs, régions d'expansion, terminaison de flots

- Connecteurs
  - *cf.* objets paramètres entrée/sortie actions
- Régions d'expansion
  - actions qui se passent pour plusieurs éléments de même type (itératif ou concurrent)



Modélisation de processus métier

- Modéliser le fonctionnement de l'organisation
- Objets responsables (stéréotypes)
  - Case worker
    - interaction avec l'ext. de l'entreprise
  - Internal worker
  - Entity
    - objet passif
- Partitions / couloirs d'activité

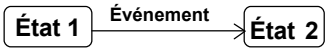


Plan du cours

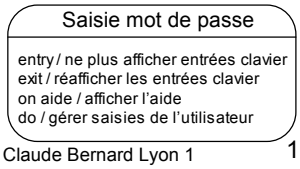
- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- Diagrammes d'interaction
- Diagrammes d'activité
- **Diagrammes de machines d'état**
- Diagrammes de composants et de déploiement
- Autres diagrammes UML
- Autres diagrammes non UML
- Autres points liés à UML

Diagrammes de machines d'états

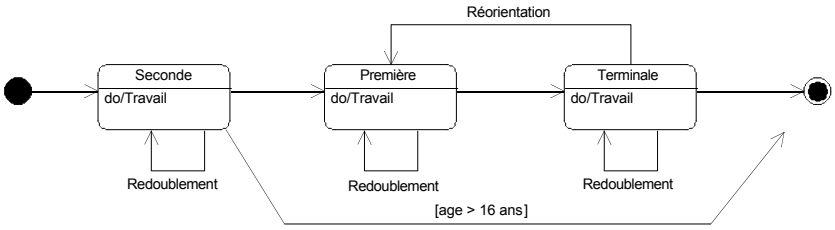
- Abstraction des comportements possibles pour une classe
  - automate à états finis décrivant les chemins possibles dans le cycle de vie d'un objet
- Etat d'un objet
  - situation nommée d'un objet qui répond à certaines conditions (durée/stabilité)



- Transition entre états
  - réponse de l'objet dans un certain état à l'occurrence d'un événement
    - passage d'un état à un autre sur événement + condition respectée,
    - action à exécuter
- Dans un état
  - activité : continue (sonnerie), tâche de fond (pagination), attente, suite d'actions…



Exemple



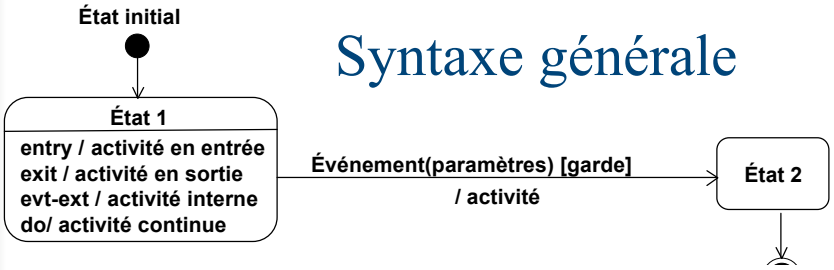
Petit exercice à faire en classe

- Tracer un diagramme de machines d'états pour un objet « TP-Etudiant ».

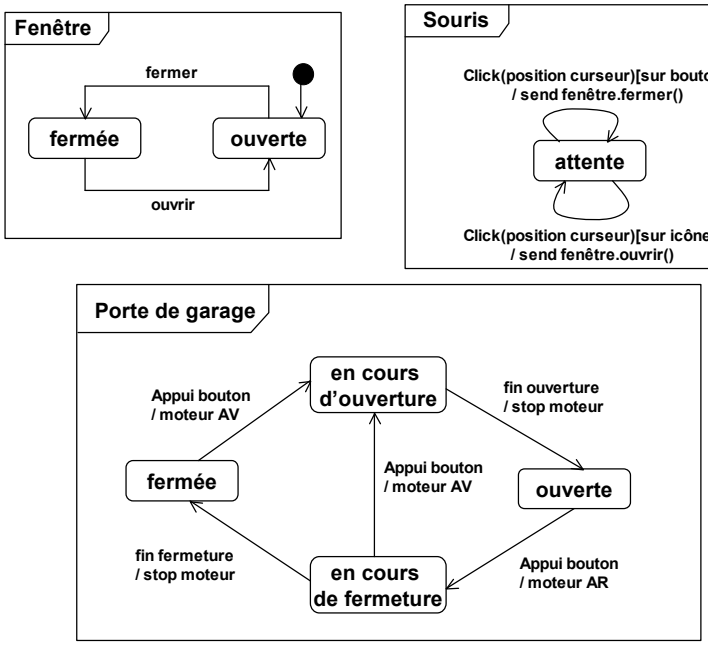
Utilisation

- Pour se concentrer sur le fonctionnement d'une classe
  - décrire / fixer le comportement concret de la vie d'une objet
    - lié à un ou plusieurs scénarios
- Pour les classes complexes
  - objets réactifs complexes (objets métier…)
  - protocole et séquences légales (sessions…)
  - en général pas plus de 10% des classes d'une application
    - plus en télécommunication / moins en informatique de gestion
- Larman
  - navigation dans un site web, IHM
    - enchaînement de pages/fenêtres

Syntaxe générale

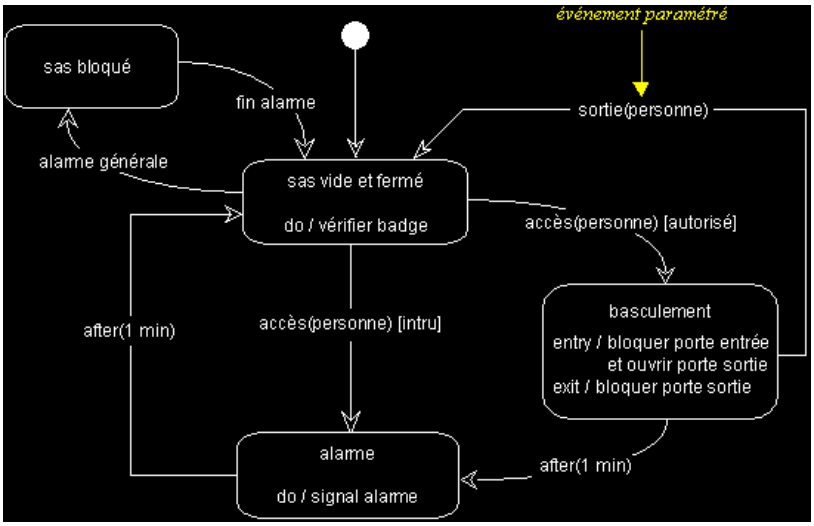


- Transition
  - tout est facultatif mais absence d'événement rare
  - événements
    - résultants de messages entre objets
    - internes : when(maximum atteint)
    - temporels : after(3 jours)
  - activité classique : envoyer un message à une cible
    - send cible.message(arguments)
- État
  - activités internes (ordinaires) : instantanées
    - autotransition sur événement extérieur, instantané
    - deux activités spéciales : sur entrée et sortie
  - activités continues : peuvent être interrompues
    - do / activité



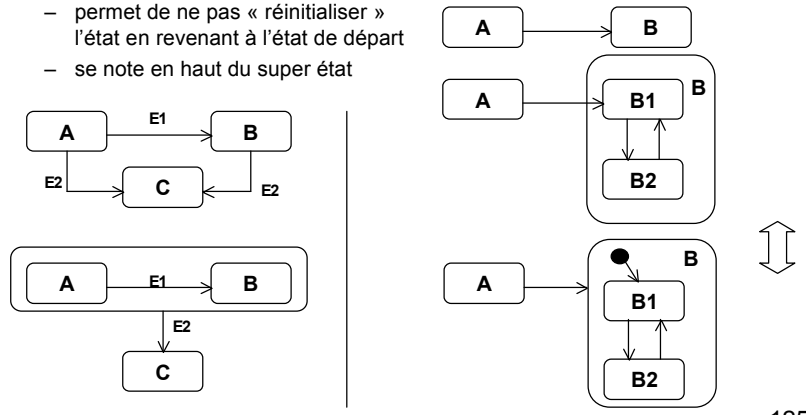
http://uml.free.fr/cours/p20.html

Autre exemple

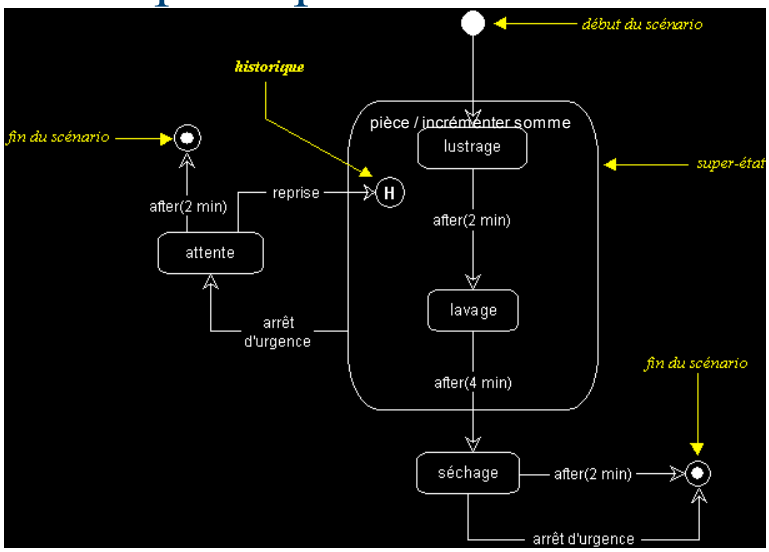


Super-états / états composites

- Pour factoriser un comportement
  - transitions déclenchées par le même événement, conduisant au même état
- Transition interne
  - couple événement / activité sans effet sur l'état courant
  - permet de ne pas « réinitialiser » l'état en revenant à l'état de départ
  - se note en haut du super état

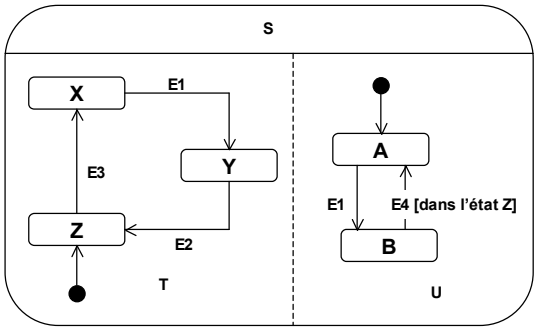


Exemple super-état



États concurrents

- Pour décomposer des états complexes
- Exercice : trouver le diagramme d'état « à plat » équivalent



Implémentation

- Utilisation de `case … switch`
  - déconseillé
- Utilisation du pattern état
  - arborescence de classes états
  - délégation de la gestion de l'état
- Tables d'états
  - représentation tabulaire du diagramme
    - état source, cible, événement, garde, procédure à exécuter
  - permet de « paramétrer » le comportement de la classe

Plan du cours

- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- Diagrammes d'interaction
- Diagrammes d'activité
- Diagrammes de machines d'état
- **Diagrammes de composants et de déploiement**
- Autres diagrammes UML
- Autres diagrammes non UML
- Autres points liés à UML

Pour chaque type de diagramme

- Présentation générale
  - Syntaxe fondamentale
  - Exemples simples
  - Petit exercice à faire en classe
- Utilisation
  - Quand et pourquoi utiliser ? (grands types d'utilisations / moment par rapport à la conception).
- Présentation avancée
  - Implémentation (si cela signifie quelquechose)
  - Plus loin dans la syntaxe
  - Choses à savoir

Diagrammes de composants

- Objectif
  - représenter l'organisation et les dépendances entre composants logiciels
  - description des composants et de leurs relations dans le système en construction
- Composant
  - partie physique et remplaçable d'un système qui se conforme à et fournit la réalisation d'interfaces
  - doit être compris comme un élément qu'on peut acheter, associer à d'autres composants (*cf. HiFi)*
  - division en composants
    - décision technique *et* commerciale (Fowler)
- Remarque
  - UML1 : composant = n'importe quel élément, y compris fichiers.
  - UML2 : utiliser les *artefacts* pour représenter des structures physiques (jar, dll…)

(Fowler 04)

Diagramme de composants

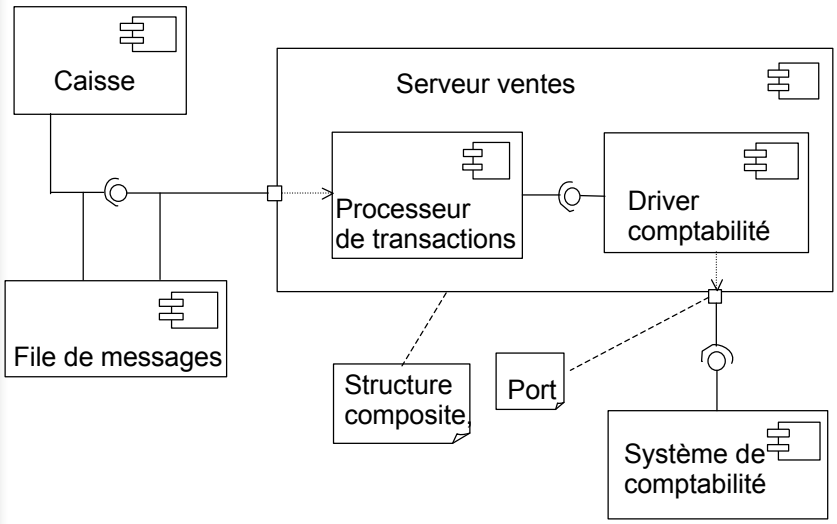


Diagramme de déploiement

- Disposition physique des différents matériels qui entrent dans la composition d'un système, ainsi que disposition des programmes exécutables sur ces matériels.
  - visualiser la distribution des composants dans l'entreprise
  - unités = nœuds
    - équipements = matériel
    - environnement d'exécution = logiciel
  - un noeud contient des artefact : classes, ...
- Relations entre éléments : supports de communication

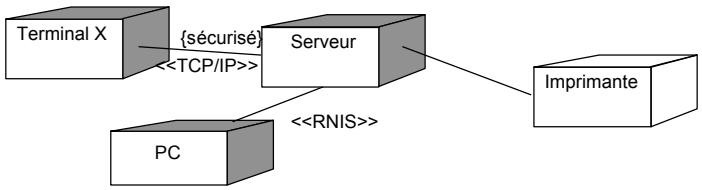
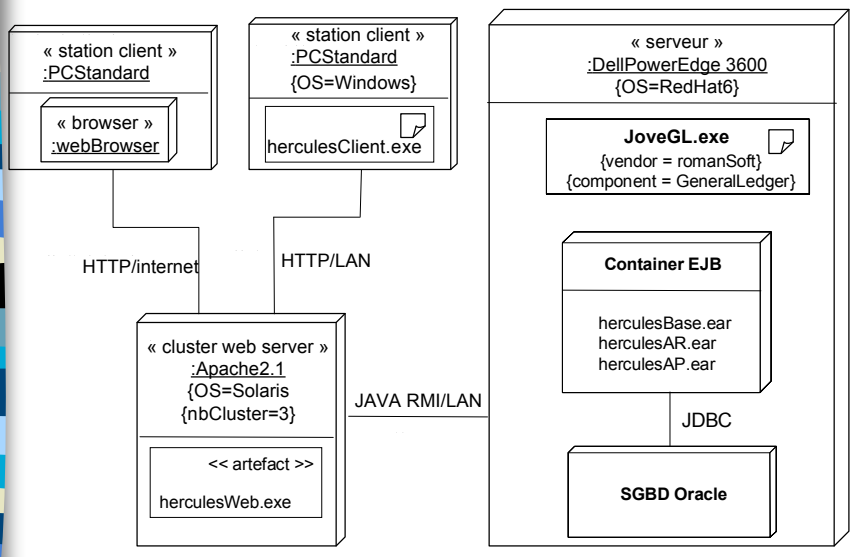


Diagramme de déploiement



Plan du cours

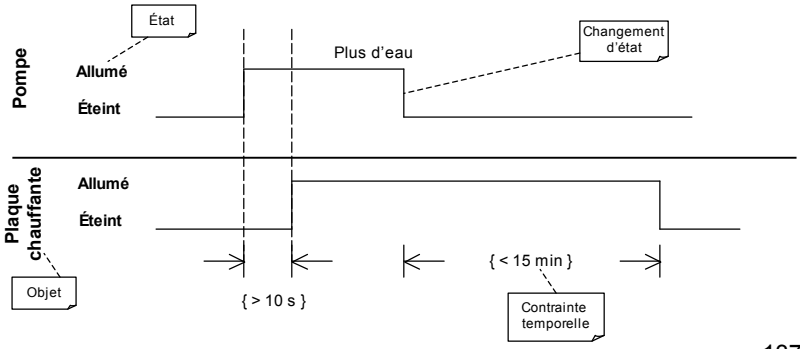
- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- Diagrammes d'interaction
- Diagrammes d'activité
- Diagrammes de machines d'état
- Diagrammes de composants et de déploiement
- **Autres diagrammes UML**
- Autres diagrammes non UML
- Autres points liés à UML

Vue d'ensemble des interactions

- Mixte diagramme activité / diagrammes de séquences
  - les actions sont remplacées par des diagrammes de séquence
- Trop tôt pour juger de l'utilisation / utilité effective

Diagramme de timing

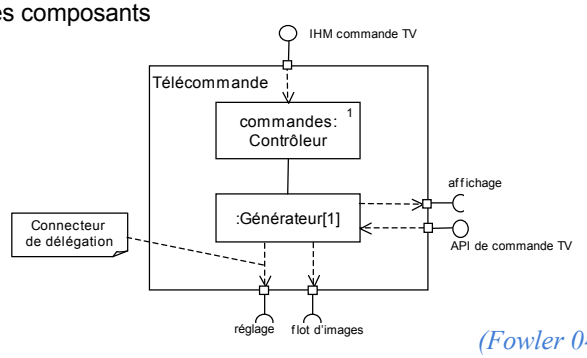
- Interactions avec focus sur les changements d'états d'objets et les contraintes temporelles associées
  - ligne de vie horizontale
- Utilisé surtout dans les applications temps réel



(Fowler 04)

Structures composites (classeurs structurés)

- Décomposer structurellement une classe
  - parties, connecteurs
- Montrer les réalisations / utilisations d'interfaces
  - ports, interfaces
- Adaptés pour les composants



(Fowler 04)

Diagrammes de collaborations

- Non officiels dans UML2, se rapprochent des diagrammes de structure composite
- Permettent de présenter les éléments impliqués dans une collaboration, et le rôle qu'ils y jouent
  - fixer les éléments et les rôles pour les diagrammes d'interaction
- En théorie utilisés pour représenter des *patterns*

Plan du cours

- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- Diagrammes d'interaction
- Diagrammes d'activité
- Diagrammes de machines d'état
- Diagrammes de composants et de déploiement
- Autres diagrammes UML
- **Autres diagrammes non UML**
- Autres points liés à UML

Diagrammes de contexte (Roques, 2004)

- Diagramme de contexte statique
  - diagramme de classe
    - une classe système
    - tous les acteurs autour
- Diagramme de contexte dynamique
  - diagramme de communication qui résume les messages entre système et acteurs (pas de numérotation)
- Diagramme de contexte statique étendu
  - diagramme de contexte statique avec
    - Attributs et opérations de hauut niveau pour le système et les acteurs non humains
- Remarque
  - « diagrammes de classes avec messages »
    - diagramme de classe avec résumé des messages entre classes

Diagramme de flux d'écrans informel (Fowler, 2004)

- Un rectangle par écran
- Des flèches pour la navigation
  - éventuellement un nom signifiant le lien

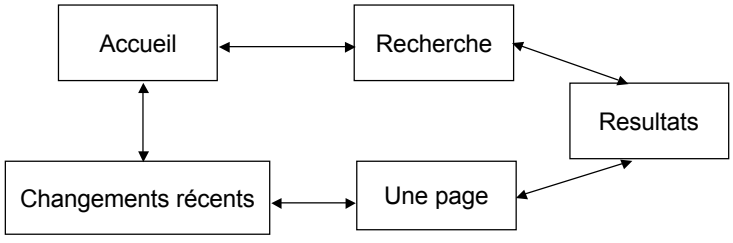


Table de décision (Fowler)

- Pour représenter des conditions logiques complexes
- Deux parties
  - conditions
  - conséquences

Client privilégié	X	X	O	O	N	N
Commande prioritaire	O	N	O	N	O	N
International	O	O	N	N	N	N
Prix	150	100	70	50	80	60
Alerte	oui	oui	oui			

Cartes CRC

Classe	
Responsabilite1	Coll1,2

- Classes - Responsabilités - Collaborateurs
  - à la base inventé pour l'enseignement
- Jouer des scénarios avec des cartes
  - 5-6 participants
- Une carte
  - nom de classe
  - tableau à deux colonnes
    - responsabilité de la classe : quelque chose d'un objet doit faire
    - collaborateurs : classes avec lesquelles il faut collaborer pour assurer la responsabilité
- Jeu
  - déterminer des classes de départ avec responsabilités évidentes
  - jouer les scénarios, ajouter les responsabilités, créer de nouvelles classes, *etc.*
- Voir par exemple
  - http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/

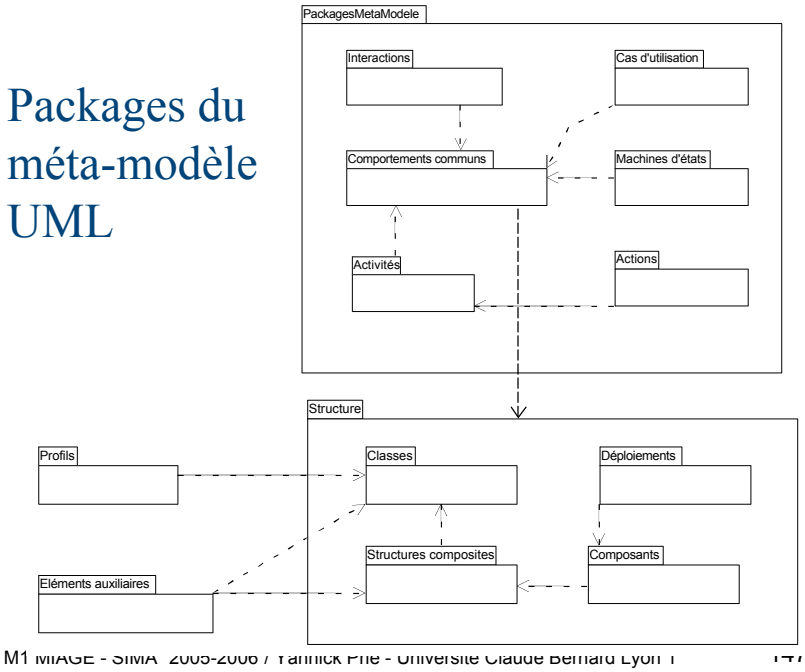
Plan du cours

- Introduction à UML
- Généralités sur la notation
- Diagrammes de classes, objets, packages
- (Diagrammes de) cas d'utilisation
- Diagrammes d'interaction
- Diagrammes d'activité
- Diagrammes de machines d'état
- Diagrammes de composants et de déploiement
- Autres diagrammes UML
- Autres diagrammes non UML
- **Autres points liés à UML**

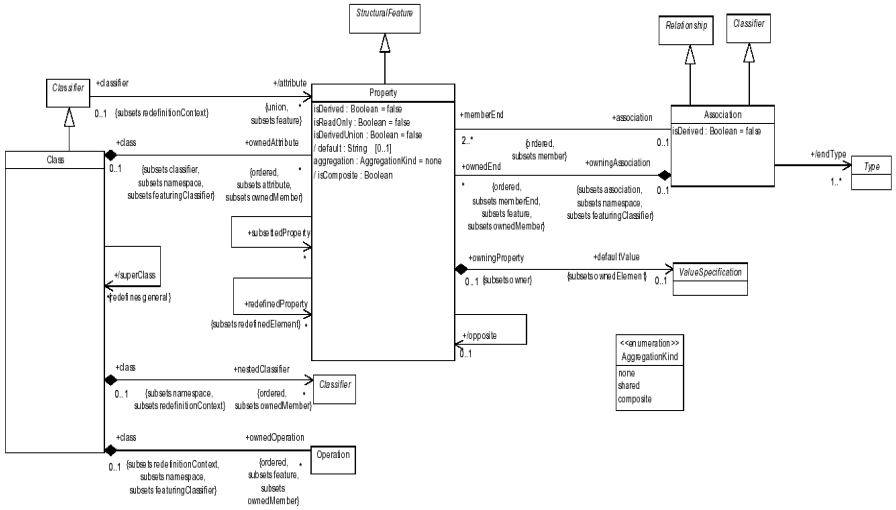
Méta-modèle

- L'ensemble de UML est décrit en UML
- Méta-modèle UML
  - description formelle de tout ce qu'il est possible de construire et de la sémantique associée
  - nécessaire pour les fabricants d'outils
- Essentiellement
  - diagrammes de classes avec contraintes et description de la signification dynamique des éléments
- Remarque : MOF (Meta Objet Facility)
    - Méta-méta-modèle permettant de décrire UML
    - mais aussi CWM
      - Common Warehouse Metamodel : structure de BD

Packages du méta-modèle UML



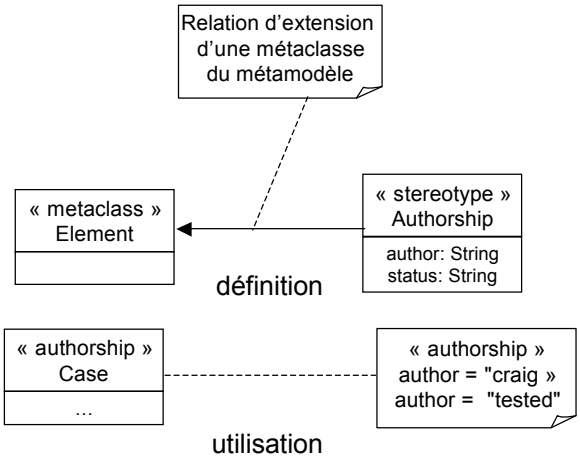
Extrait du méta-modèle



D'après OMG UML2 Superstructure, Figure 30

(Larman, 2005)

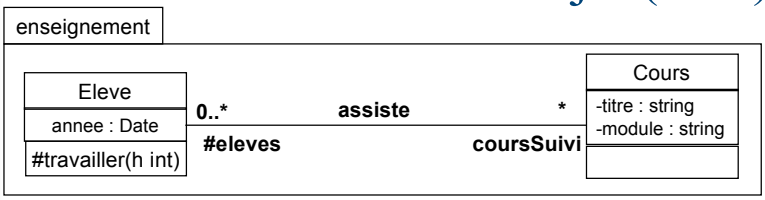
Extension d'UML : stéréotypes



Modèle UML / programme OO

- Modèle
  - classes
    - attributs, associations
    - opérations
    - spécialisation
  - packages
  - interactions
    - séquences de messages entre objets
- Possibilité de traduire tout ou partie du modèle dans un langage
  - objet ; Java, C++, C#, *etc.*
  - relationnel : classes, attributs

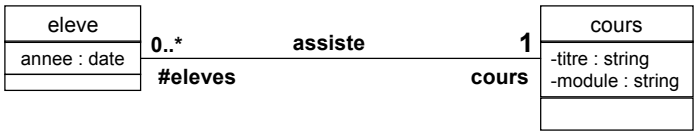
Traduction en objet (Java)




```
package enseignement ;
public class Eleve
{
    private Date annee ;
    private Court coursSuivi[] ;
    public Eleve() ;
    public Date getAnnee()
    {
        return annee ;
    }
    public void setAnnee(Date uneAnnee)
    {
        annee = uneAnnee ;
    }
    …
```

```
…
public associerCoursSuivi(unCours :
Cours)
{   /* ecrire ici */
}
public nbCoursSuivi
 {
     return coursSuivi.length ;
}
protected travailler(h :  int)
{
     /* ecrire ici */
}
/* etc. */
}
```

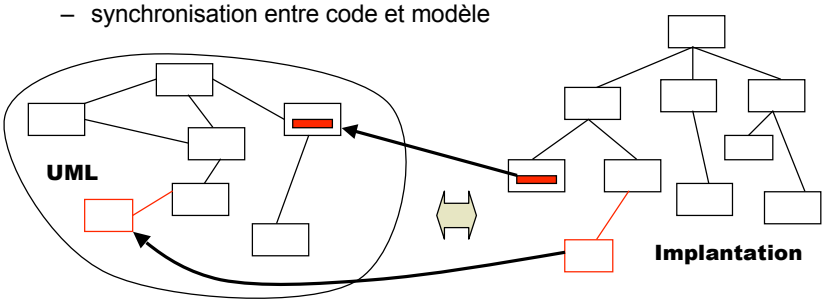

Traduction en relationnel




```
CREATE TABLE eleve (
    eleve_id NUMBER (5) ,
    annee DATE,
    PRIMARY KEY (eleve_id)
) ;
CREATE TABLE cours (
    eleve_id NUMBER (5)  REFERENCES eleve(eleve_id) ,
    cours_id NUMBER (5) ,
    titre CHAR (128) ,
    module CHAR(48) ,
    PRIMARY KEY (cours_id)
) ;
```

Ingénierie UML

- Pro-ingénierie
  - générer le code à partir du modèle
  - outils : paramétrage (héritage, associations…)
- Rétro-ingénierie
  - générer le modèle à partir de l'implantation
  - seuls les outils automatiques peuvent le faire
- Ingénierie bidirectionnelle (roundtrip engineering)
  - synchronisation entre code et modèle



Outils UML

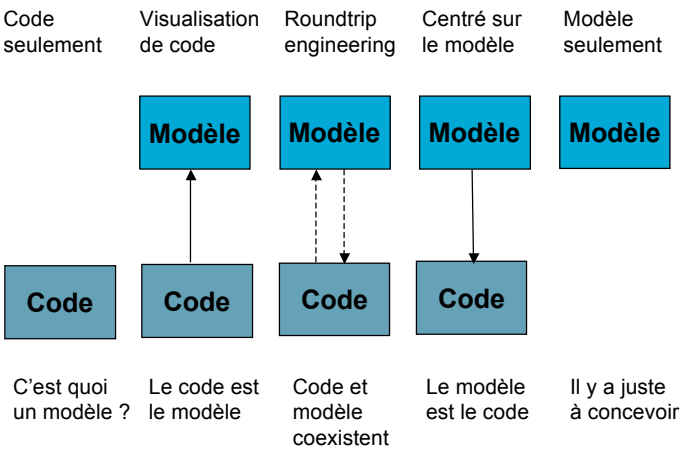
- Outils de dessins améliorés
  - intègrent les diagrammes comme simples formes
- Outils UML
  - gestion des diagrammes et du modèle
    - Vérification de cohérence
  - génération de code
    - squelettes de classes / contenus des méthodes (peu)
  - rétro-ingénierie
    - diagrammes de classes
    - diagrammes de séquences (peu)
  - de plus en plus intégrés / en compléments d'autres outils
    - IDE, gestion de projet, du risque, des besoins, de la qualité, des tests, du workflow, *etc*.

XMI

- XML Metadata Interchange
- Pour échanger des modèles UML entre outils
- Utilisation d'une syntaxe XML
  - eXtensible Markup Language
- Remarque
  - génération de documentation HTML
    - transformation XSL
  - diagrammes
    - transformation en SVG (Standard vector Graphics)

D'après http://www-128.ibm.com/developerworks/rational/library/3100.html

Modèles et code

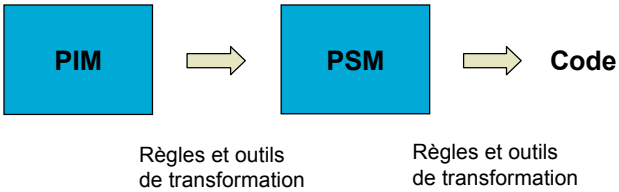


MDA : Model Driven Architecture

- Architecture pilotée par les modèles
  - mis en place et supporté par l'OMG
    - http://www.omg.org/mda/
  - UML comme langage de programmation
  - passer d'un développement centré sur le code à un développement centré sur les modèles
  - MDD (Model Driven Development)
    - terme plus général, non OMG
- Pour permettre, de la façon la plus intégrée possible
  - productivité
  - portabilité
  - interopérabilité
  - maintenance
  - documentation

MDA

- Deux types de modèles
  - PIM (Platform Independant Model)
    - en UML
  - PSM (platform specific model)
    - pas obligatoirement en UML



MDA : conclusion

- Pour certains
  - l'avenir de l'informatique
  - des outils existent
  - des « leaders » utilisent et tentent de promouvoir
- D'autres sont moins convaincus
  - entre
    - « ça ne marchera jamais »
    - « je demande à voir »
- Acceptation
  - possibilité de programmer mieux et plus vite (moins cher)
  - rapport apprentissage / gain
    - dans les entreprises
    - dans les écoles
- A suivre…

Contrainte

- Condition ou restriction sémantique associée à un ou plusieurs éléments de modèle exprimée
  - en langue
  - dans un langage formel
- Assertion qui doit être vraie
  - entre des appels d'opérations (qui changent le système)
  - à des moments précis par rapport aux appels d'opérations

OCL Object Constraint Language

- Standardisé par l'OMG
- Permet d'exprimer des contraintes de façon formelle
- Expression
  - d'invariants au sein d'un classe ou d'un type : bon fonctionnement des instances
  - contraintes au sein d'une opération : bon fonctionnement de l'opération
  - pré- et post- conditions d'opérations : avant et après l'exécution
    - *cf.* programmation par contrats (Meyer)
  - gardes : sur la modification de l'état d'un objet
  - expressions de navigation : chemins
- Utilisation
  - génération de code
  - MDA

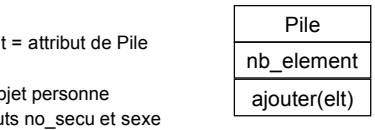
OCL : exemples

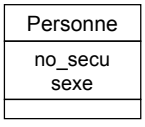
context nom_élément [**inv**|**pre**|**post**] : expression de la contrainte

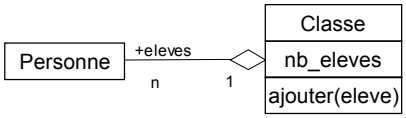
```
context Pile inv :
    self.nb_elements >= 0   -- nb_element = attribut de Pile

context Personne inv    -- intégrité de l'objet personne
                                / attributs no_secu et sexe
    if sexe = "F"  then no_secu.commence_par() = 2
    else no_secu.commence_par() = 1
    endif

context Classe::ajouter(un_eleve : eleve)
    pre classe_non_surchargée : nb_eleves <= 25
    post : eleves→exists(un_eleve)
```







Conclusions sur UML

- Propriétés d'UML
  - unification de concepts de modélisation
  - puissance d'expression
    - nombreux formalismes (issus de méthodes existantes)
  - compromis formalisation / niveau d'abstraction / indépendance aux langages / sémantique fixée / extensibilité
- Langage universel
  - pour de multiples domaines
  - pour diverses activités de la conception
  - dans différents modes
    - esquisse, plan, programme

Conclusions sur UML

- Standard international : adopté un peu partout
  - les diagrammes sont simples, faciles à lire et à communiquer
  - beaucoup de variantes locales
  - outils puissants
    - dessin
    - pro et rétro ingénierie
    - MDA
- UML n'est qu'un langage
  - encapsule tout ou partie de la sémantique de description
  - ne dit pas comment construire les modèles
- Il faut utiliser des méthodes
  - démarches de conception et d'utilisation des diagrammes et des modèles