



UML Agile

Agile UML

zam@dauphine.fr



Descriptif

- Ce cours s'attache à
 - consolider les notions de conception à l'aide d'UML et d'un outil de modélisation,
 - à valider la capacité du futur professionnel à construire et à faire évoluer collectivement les spécifications d'une application et
 - à faire émerger une méthodologie de travail **agile**



Etapes & infos

1. Prémices méthodologiques : notation UML et processus **agiles** adaptés.
2. Rétro-conception outillée d'un système existant
3. Formulation méthodologique
4. Soutenance collective, avec notation individuelle
 - Prérequis : avoir pratiqué UML ou avoir suivi la "Mise à niveau UML »
 - $N = 100\% * PR$, points marqués [+/-] au fil de l'eau
 - Volume horaire : 24, ECTS : 3

© 2013 zam@dauphine.fr



Partie 1 - Modélisation

Modélisation microscopique : mono/multi diagrammes

- Objectifs
 - Comprendre, élaborer et raisonner par le **modèle UML** (modélisation)
 - Assimilation notation et des règles de vérification des diagrammes
⇒ le modèle = « un contrat » compris souscrits par tous les participants
- Déroulement
 - Vision objet et modélisation graphique
 - Diagramme par diagramme (classe, états, cas d'utilisation et activité)
 - Exemples, notation, exercices mono et multi diagrammes
 - Papier et outillage
 - Survol des diagrammes restants



© 2013 zam@dauphine.fr



Partie 2 – Repères agiles

- *Est-ce une méthodologie ?*
- *Agile*
 - « réactif aux changements, sans compromis en qualité »
- *Démarches agiles*
 - *RAD, Lean, DSDM, Scrum, Kanban, FDD, XP, ...*
- *Illustrations*

© 2013 zam@dauphine.fr



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value :

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right,
we value the items on the left more.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler,
James Grenning, Jim Highsmith, **Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin,**
Steve Mellor, **Ken Schwaber, Jeff Sutherland, Dave Thomas**

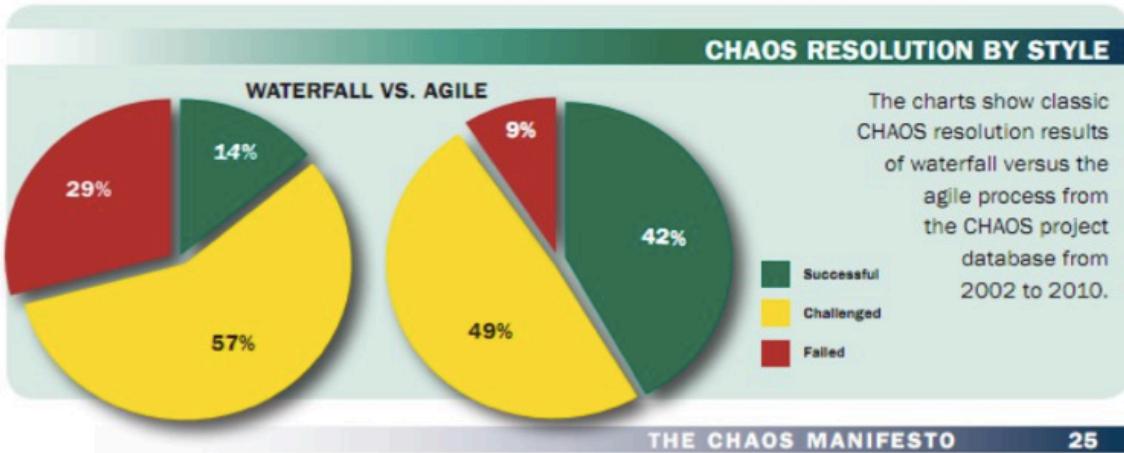
© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

<http://agilemanifesto.org/>

© 2013 zam@dauphine.fr



Cascade vs Agile



© 2013 zam@dauphine.fr



Partie 3 – Rétro-Modélisation

Étude de cas *macroscopique et documentée*

- Objectifs
 - Comprendre, élaborer et raisonner par l'expérience et le modèle
 - Assimilation de la méthodologie et des règles de vérification
 - ⇒ Le modèle = « un contrat » compris souscrits par tous : User & Dev
- Entrée
 - Un système existant, à analyser, puis à rétro-modéliser
- Sortie
 - Projet UML/Outil
 - Modèle (ensemble de diagrammes corrélés) légèrement commenté
 - Proposition méthodologique : démarche agile modélisée

© 2013 zam@dauphine.fr



Plan (I)

I. Introduction

- Vision objet et modélisation graphique avec UML
- UML - concepts clé et survol des diagrammes

II. Modélisation objet, diagramme par diagramme

- Classe, états, cas d'utilisation et activité
- Exemples, notation, exercices
- Mono et multi diagrammes
- Papier et outillage

III. Survol des diagrammes restants

- Séquence & collaboration
- Composantes & déploiement

© 2013 zam@dauphine.fr



Le paradigme orienté objet

• Vision du monde

- applicable à la fois au développement et à la modélisation ...
- principal objectif : **la simplification** des applications très complexes

© 2013 zam@dauphine.fr



Les années 60 : la simulation

- Le but
 - observation du comportement global d'un système à partir de l'interaction d'un nombre important d'éléments autonomes bien connus
 - ex : aéroport, trafic routier
- Le problème
 - nombreux réglages nécessaires pour la régulation du fonctionnement global : remises en cause

© 2013 zam@dauphine.fr



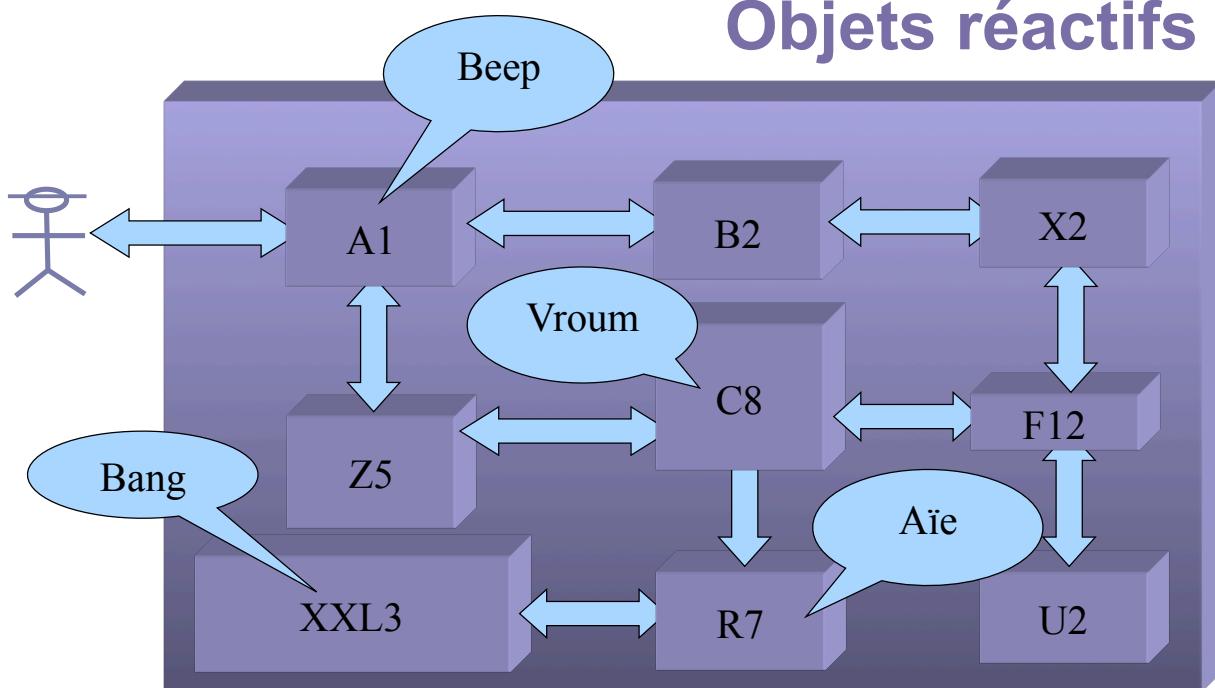
Les années 60 : la simulation

- La solution
 - univers : ensemble d'entités distinctes => objets
 - objet : caractérisé par propriétés et comportement
 - algorithme global unique et simple

© 2013 zam@dauphine.fr



Objets réactifs



© 2013 zam@dauphine.fr

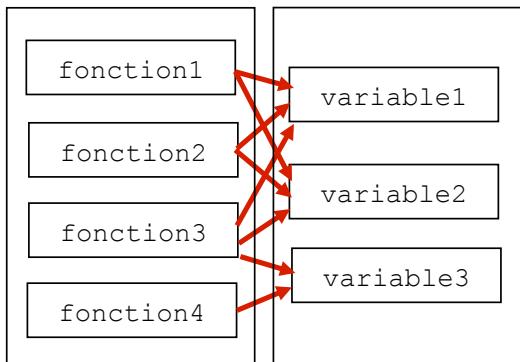


Avantages

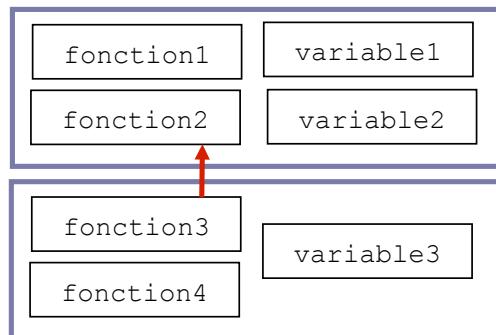
Maintenable et évolutif

- Approche procédurale vs. approche OO
 - Encapsulation
 - regroupement thématique donnée + traitement
 - réduction du nombre de paramètres => complexité--

Paradigme procédural



Paradigme orienté objet

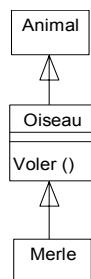


- Héritage : nouvel attribut NoSS
=> traitements généraux non remis en cause

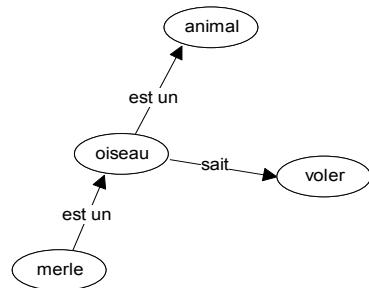
© 2013 zam@dauphine.fr



Approche intuitive



Graphe d'héritage



Réseau sémantique

« L'oiseau, vole-t-il ? »
« Le merle, vole-t-il ? »



Exercices

- Donnez quelque exemples de hiérarchies de classes d'objets.
- Comparez la taille occupée en mémoire par Toto et Dupont.
- Le merle, vole-t-il ?



Cycle de vie d'un projet

- Phases et activités (simplification)
 - Analyse, Conception, Développement, Test
 - Test : exécution
 - Exécutable + autres ressources (BD, fichiers)
 - Développement : programmation
 - Sources : PascalObjet, C++, Java, C#, ...
 - Analyse : compréhension du problème
 - Cahier de charges
 - Conception : ébauche de la solution
 - Modèle

© 2013 zam@dauphine.fr



De la programmation à la modélisation orientée objet

- Abstraire = occulter des informations jugées de moindre importance pour en garder les infos les « plus importantes »
- Ex. Structure d'un objet => classe
- Syntaxe du langage vs. symboles
 - Indépendance du langage de programmation
- Code -> diagrammes : aspects multiples et complémentaires
 - Structurel/dynamique/temporel, individuel/collectif, besoins/solution, métier/technique, conceptuel/implémentation/déploiement ...

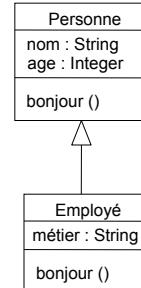
© 2013 zam@dauphine.fr



Du code au diagramme

```
class Personne {
    String nom;
    Integer age;
    void bonjour() { ... };
}

class Employé extends Personne
    String métier;
    void bonjour() { ... };
end;
```



© 2013 zam@dauphine.fr



Bilan



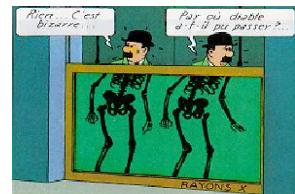
- Une application OO fonctionne grâce à l'interaction de nombreux **objets**
- Les objets sont porteurs d'information et responsables des traitements (en direct ou par délégation)
- Ils sont fabriqués à l'exécution en mémoire à partir des **classes**
- Les classes sont **programmées** par les développeurs
- Le chemin de la traduction des besoins (exigences) en classes => **modélisation**

© 2013 zam@dauphine.fr



Ce que nous captions dans un modèle

- Décrire les objets / classes
 - Anatomie et physiologie des objets
 - Histoire individuelle
 - Interactions & histoire collective
 - Photos de famille



© 2013 zam@dauphine.fr



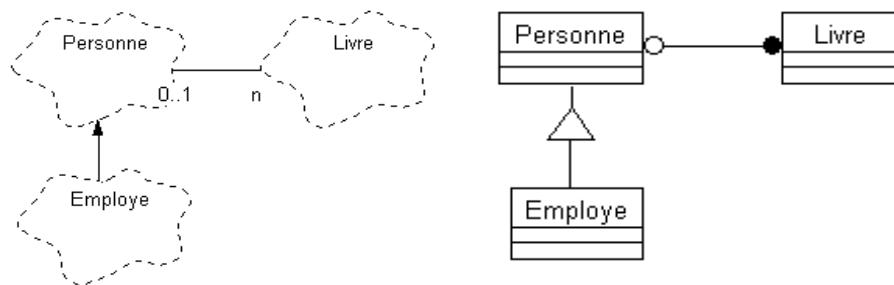
UML

Genèse, concepts clé et survol des diagrammes



Genèse d'UML...

- 80 's : la POO quitte les labos
 - Smalltalk, C++
- fin 80 's : les premières méthodes
 - Shlaer - Mellor, Coad - Yourdon, Beck - Cunningham, Booch, Rumbaugh, Odell - Martin, Jacobson, ...
 - concepts ≈ / notations ≠ , compétition, sectes



© 2013 zam@dauphine.fr



...

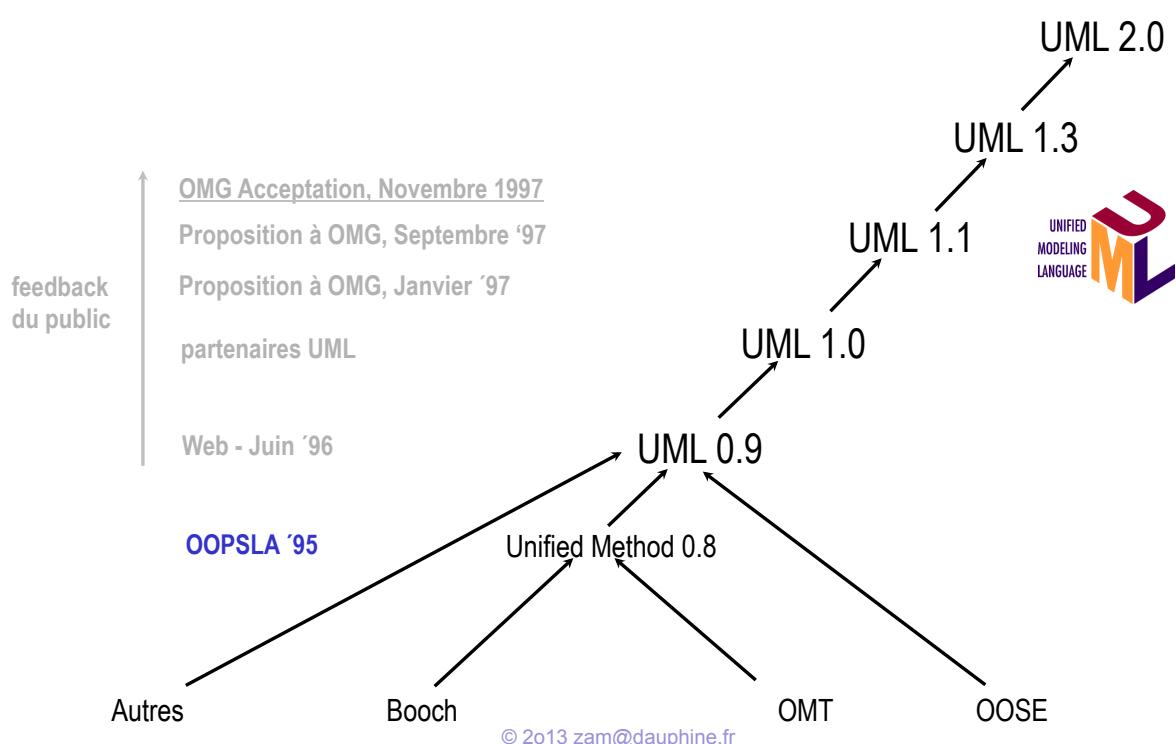
- début 90 's : standardisation ?
 - méthodologiste vs terroriste ;o)
- 94 : la méthode unifiée
 - Rumbaugh rejoint Booch chez Rational
 - « fusion » des 2 méthodes
 - « the war is over, we won »

© 2013 zam@dauphine.fr



- 96 : UML
 - Jacobson rejoint Rational
 - les trois *amigos*
 - méthode unifiée => UML
- 96 : l'OMG s'en mêle
 - groupe de travail (Odell)
 - pas un standard imposé par Rational !
- 97 : enfin un standard
 - UML (v 1.0) est proposé à l'OMG...
 - Adopté comme standard (v 1.1)

© 2013 zam@dauphine.fr



© 2013 zam@dauphine.fr



Un langage de modélisation OO

Unified Unique / standard

Modeling Construction de modèles OO

Language mots / def / grammaire

© 2013 zam@dauphine.fr



UML : principes fondateurs

- Basé sur le paradigme orienté objet
 - Encapsulation, Héritage, (Polymorphisme)
- Système
 - Ce que l'on veut modéliser (application/organisation)
 - Une communauté harmonieuse d'éléments autonomes qui interagissent pour accomplir un but commun : comportement global souhaité/observé
- Modèle
 - Une représentation du **système**
 - Concepts : objet, classe, attribut, acteur, activité, ...
- Vue
 - Une projection spécialisée du **modèle** sur un axe particulier : statique/dynamique, individuel/collectif
 - Diagrammes => symboles graphiques / objets, classes ...

© 2013 zam@dauphine.fr



Contenu du standard

- Notation
 - éléments graphiques / sémantique
 - mots / définitions
- Métamodèle
 - grammaire
 - rigueur vs intuition
- Pas de processus standardisé

© 2013 zam@dauphine.fr



Avantages escomptés

- Une meilleure vue sur le code et sur le fonctionnement de l'application : **aspect graphique/visuel**
 - communiquer avec les experts du domaine (standard)
 - vue d'ensemble d'un projet
 - fidèle au paradigme orienté objet
- Gestion de la complexité
- Apprentissage, capitalisation et communication
- Maintenabilité, évolutivité
- Qualité : moins de bugs => coût global réduit

© 2013 zam@dauphine.fr



Conclusion

UML **autorise** ces avantages sans les **imposer**

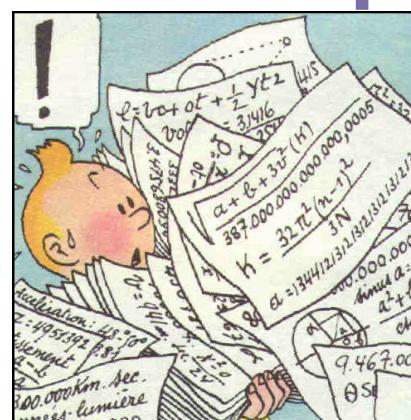
- Réserves
 - Perfectible, spécifications arides
 - nécessite : outillage, matériel didactique et expérience
 - nombreuses sources complémentaires de qualité inégale
 - n'apprend pas la modélisation
- Notre approche
 - en "douceur", progressive, par des exemples
 - sélection judicieuse des éléments essentiels
les plus utiles dans la modélisation métier (études)
 - concrète : avec un outil
 - ascendante : « bottom-up »

© 2013 zam@dauphine.fr



UML par l'exemple

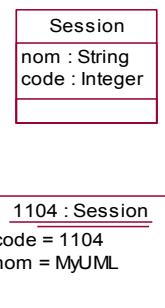
*La moitié des bugs
sont imputables
aux exigences*





Aperçu de la notation UML

- **Système**
 - Ce que l'on veut modéliser (application / organisation)
 - **Modèle**
 - Description du système
 - Concepts
 - Acteur, cas d'utilisation
 - Objet, classe, attribut ...
 - **Vue**
 - Diagrammes de ...
 - Cas d'utilisation, Classe, Objet
 - Séquence, Collaboration, État, Activité
 - Composants et Déploiement
 - Symboles
 - Icônes spécialisées : **classe, objet**



© 2013 zam@dauphine.fr



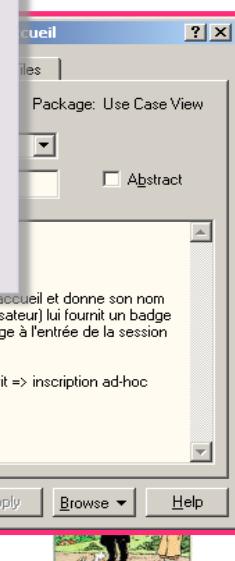
- ## **But**

 - Entrer dans la conférence

Déroulement principal

- Le **participant** se présente à l'**accueil** et donne son **nom**
 - L'**hôtesse** (système ou l'**organisateur**) lui fournit un **badge**
 - Le **participant** montre son **badge** à l'**entrée** de la **session**

Script



accueil ...

badge ...

entrée ...

inscription ...

nom ...

organisateur ...

participant

session ...

scribt => inscription ad-hoc

accueil

position

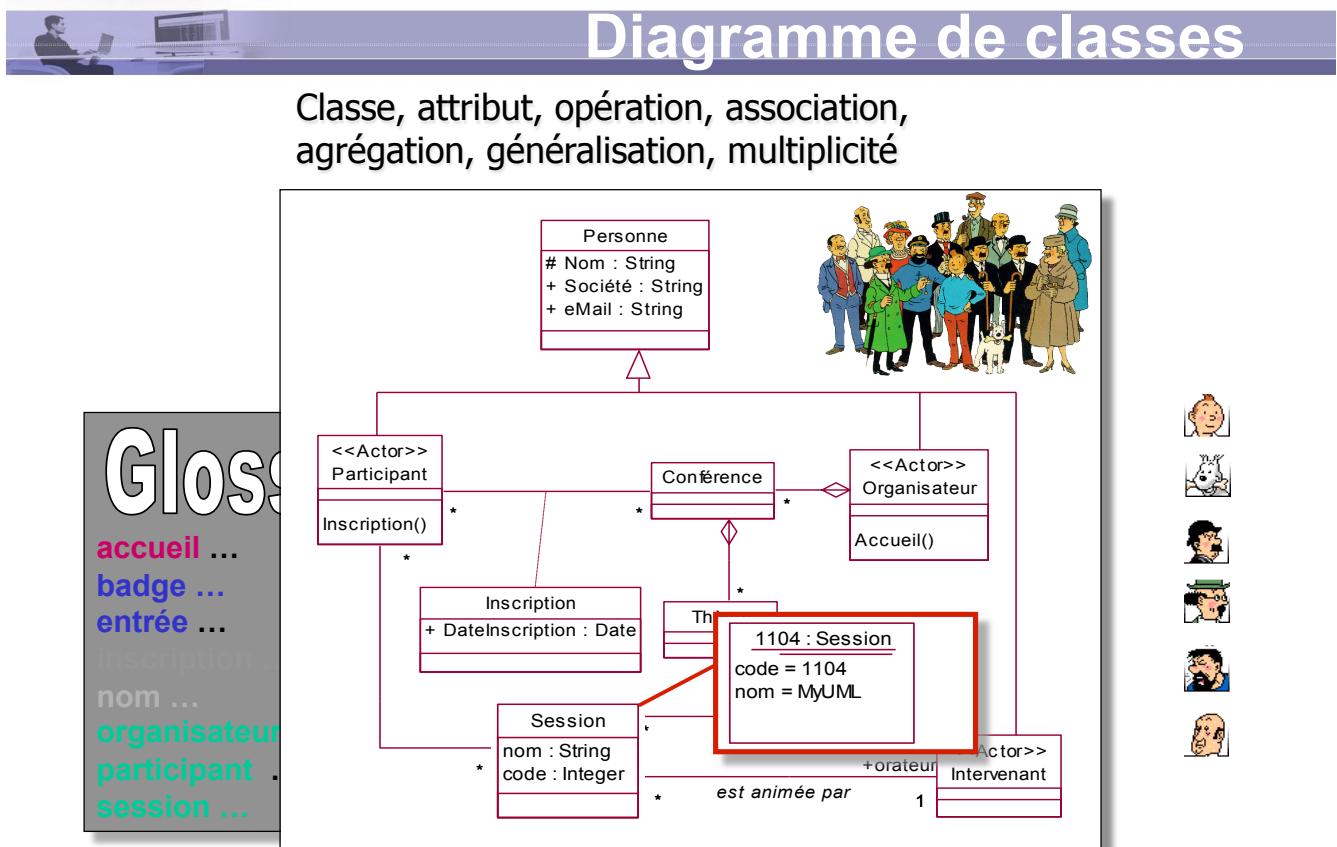
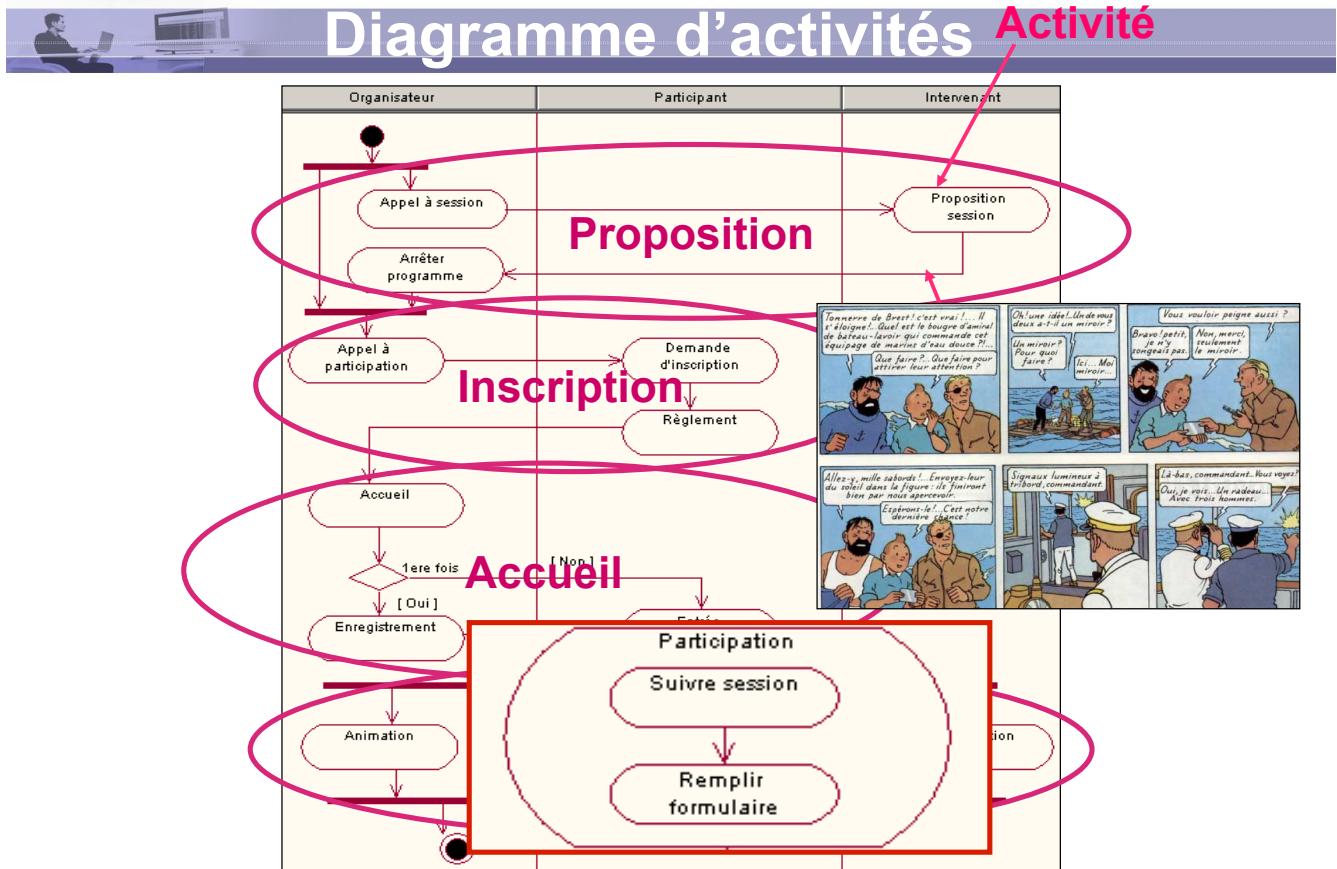
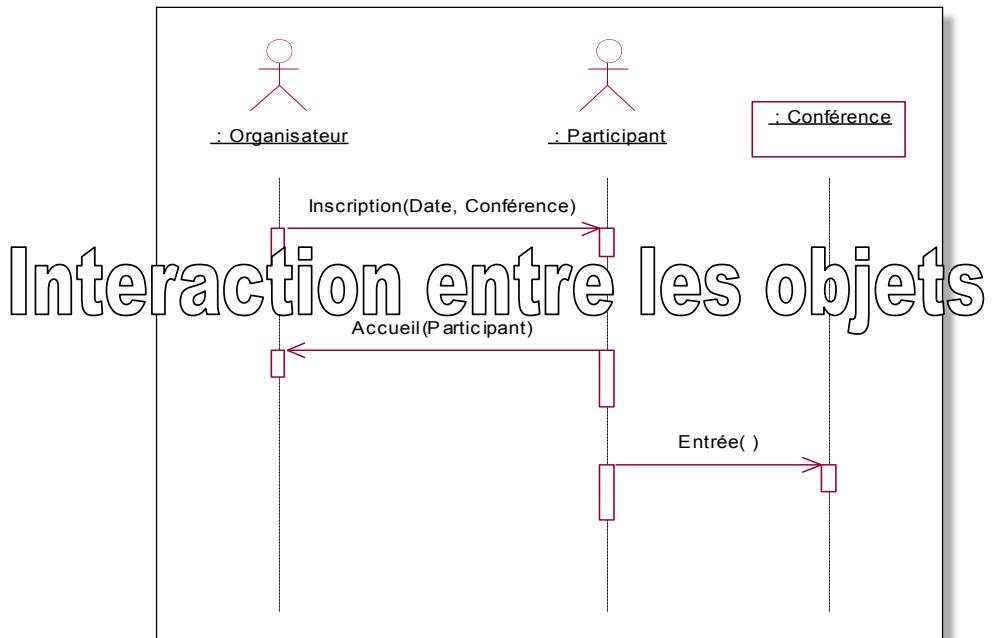




Diagramme de séquence

- Objet, ligne de vie, message, activation

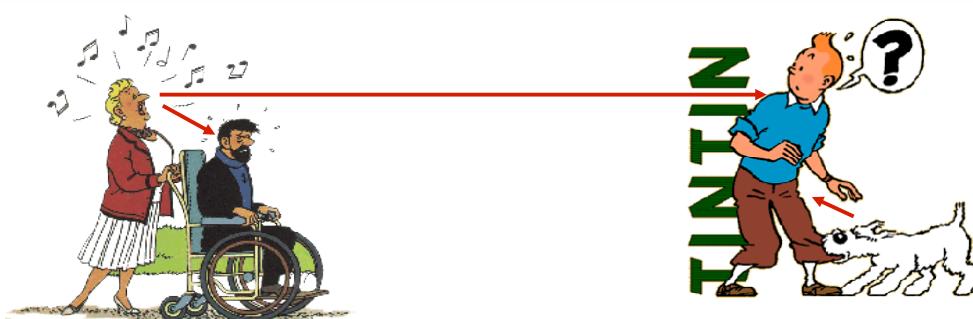
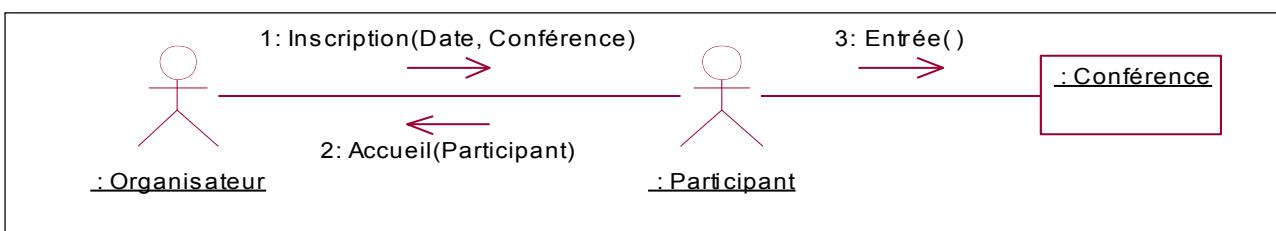


© 2013 zam@dauphine.fr



Diagramme de collaboration

- Objet, message, **lien**

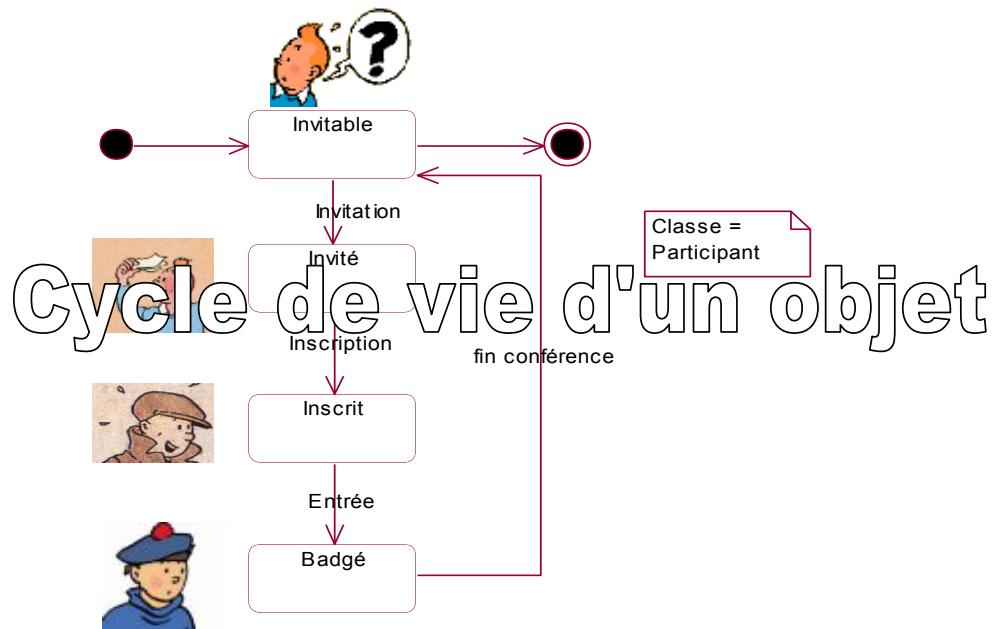


© 2013 zam@dauphine.fr



Diagramme états-transitions

- État, transition

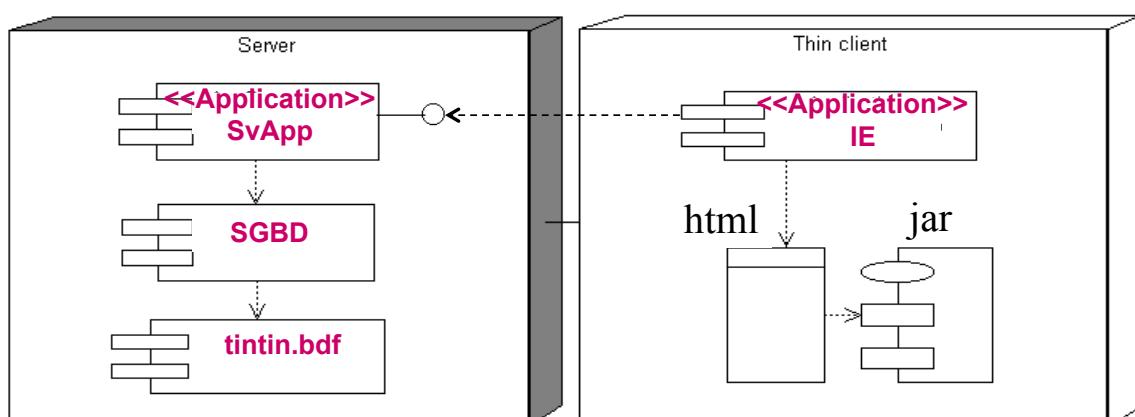


© 2013 zam@dauphine.fr



Diagrammes d'implémentation

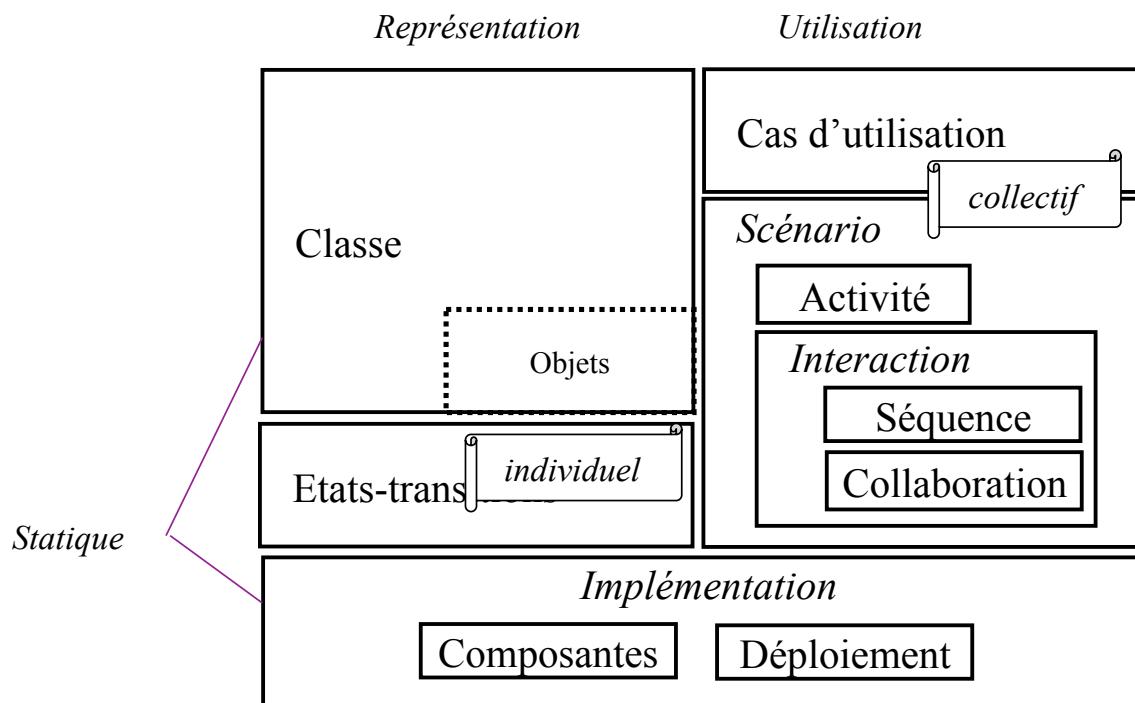
- Composants et déploiement



© 2013 zam@dauphine.fr



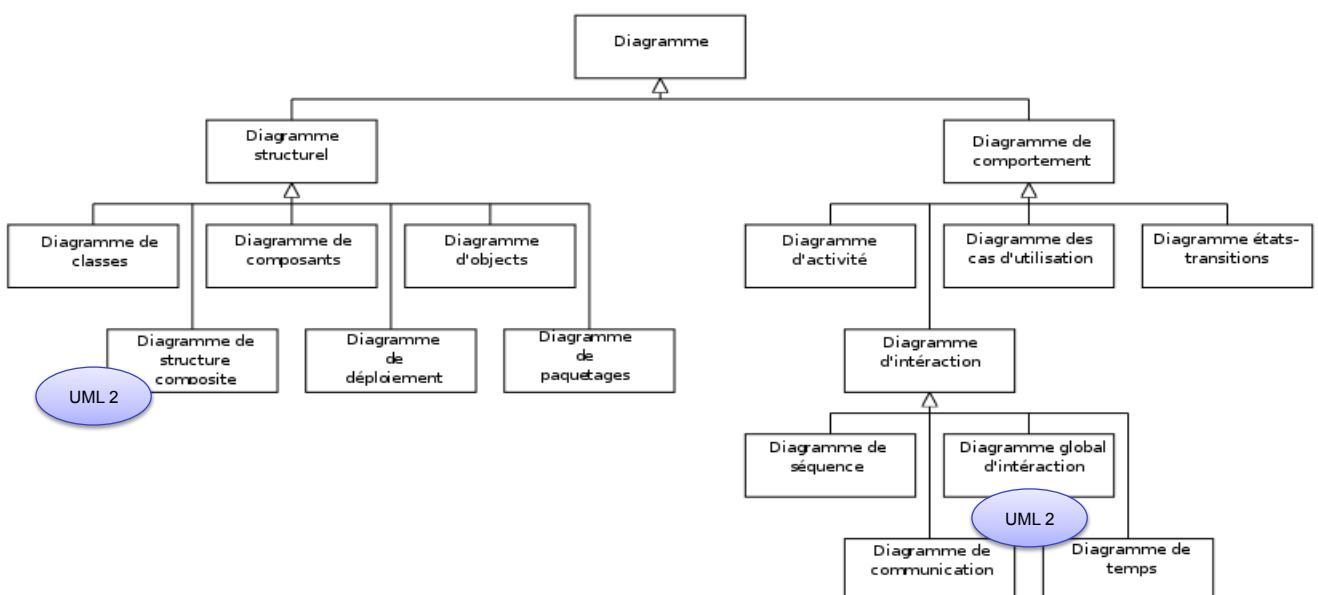
Diagrammes UML



© 2013 zam@dauphine.fr



UML 2



© 2013 zam@dauphine.fr



II. Modélisation objet

Diagramme par diagramme
(Classes, Etats, UseCases & Activités)

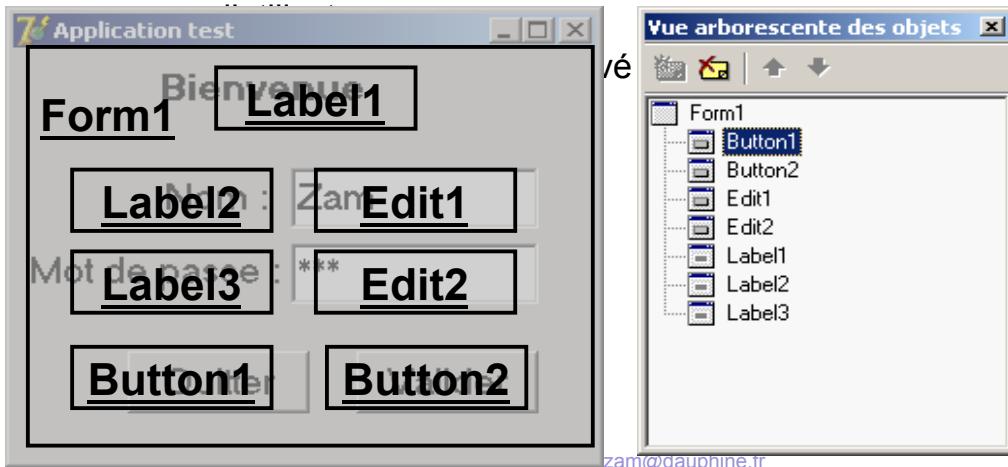


Diagramme de classes



Vision statique = anatomique

- Des objets aux modèles
 - Une application dite « orientée objet » est un assemblage d'objets qui collaborent pour réaliser les tâches exigées par

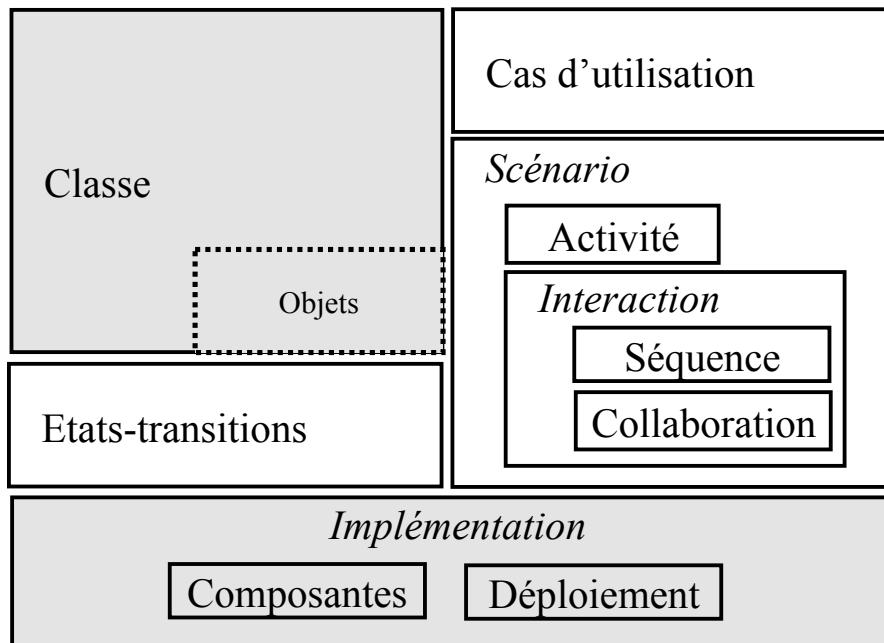


Introduction

- Particularité
 - A l'**exécution** : les objets sont fabriqués **automatiquement** à partir des usines appelées **classes** et des **données persistantes**, résidantes sur le disque dur
 - Classes : résidantes dans les .EXE, .DLL, .CLASS, .JAR
 - Données : .XML, SGBD (lecture & écriture)
 - **Développement préalable**
 - Programmation **manuelle** des classes (.PAS, .JAVA) et de la structure du SGBD (**DDL**), en pensant aux **objets**.
 - Compilation / F9 : l'IDE Delphi, C++Builder, JBuilder (javac)
 - Centaines de milliers de lignes de code, centaines de tables et colonnes
=> résultats virtuels, invisibles



Diagrammes statiques UML



© 2013 zam@dauphine.fr



Diagramme d'objets



Diagramme d'objets

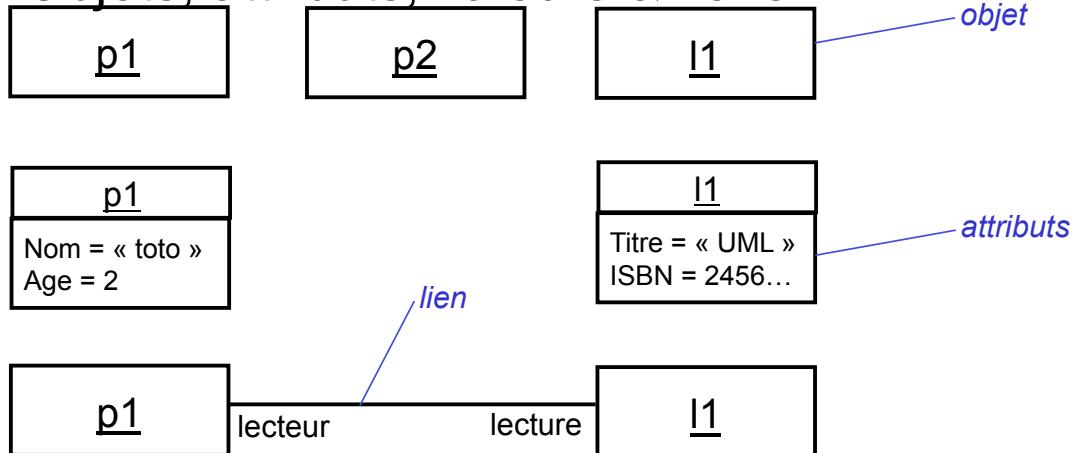
- Objectif
 - Illustrer des objets, les valeurs éventuelles de leurs attributs et leurs connections afin de pouvoir raisonner sur le fonctionnement de l'application

© 2013 zam@dauphine.fr



L'objet

- Objets, attributs, valeurs & liens



© 2013 zam@dauphine.fr



L'objet

- L'objet : n'existe qu'à l'exécution
 - Pièce élémentaire
 - Identité (adresse en mémoire)
 - Structure : ensemble d'attributs
 - (Comportement réactif)
 - Connectivité : liens avec d'autres objets
- Pour éviter le désordre
 - Attributs + comportement + liens : fournis par la classe (moule, cahier de charges, loi)

© 2013 zam@dauphine.fr



Des objets aux classes

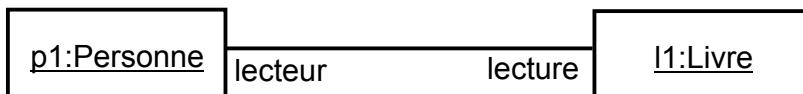
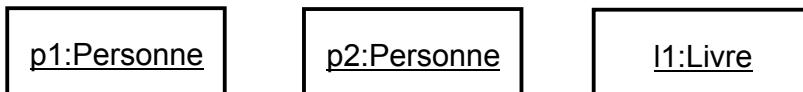
- A l'exécution
 - Une application orientée objet est constituée d'un ensemble d'objets qui collaborent pour fournir des services (interactifs)
 - Décor (IHM) : soutenu par des objets
 - Stimulus => objet interactif -> sous-traitance
- A la conception, on définit leur classes
 - Usines à objets de même type (struct. + cpt)
 - Structure : données encapsulées
 - Connexions entre objets
 - Comportement : méthodes encapsulées qui communiquent
- Objectif
 - Conception : simple, modulaire et évolutif
 - Exécution : couverture efficace des services (besoins)

© 2013 zam@dauphine.fr

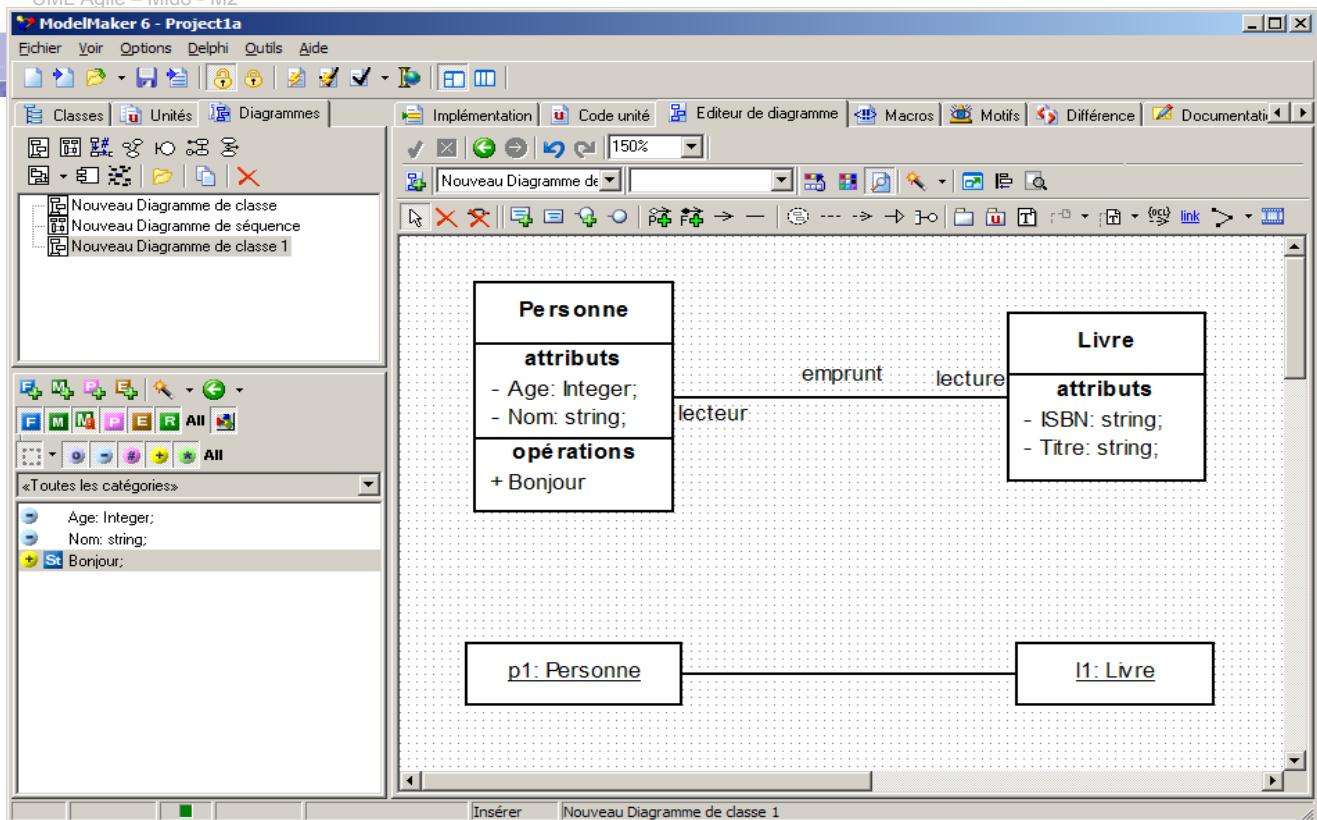


La classe

- Caractérise l'objet



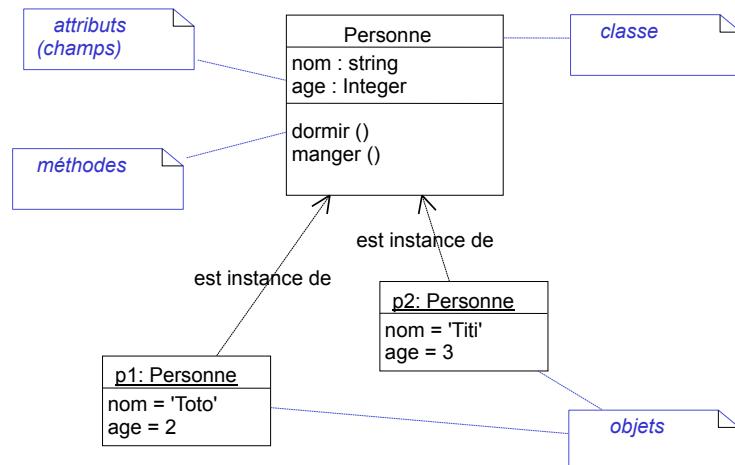
© 2013 zam@dauphine.fr



© 2013 zam@dauphine.fr



Objets et classe



© 2013 zam@dauphine.fr



Diagramme de classes



Diagramme de classes

- Objectif
 - Représenter la structure statique des objets qui composent un système à travers des classes et des relations entre les classes.
 - Typer les objets
 - Structure
 - (Comportement)
 - Connectivité

© 2013 zam@dauphine.fr

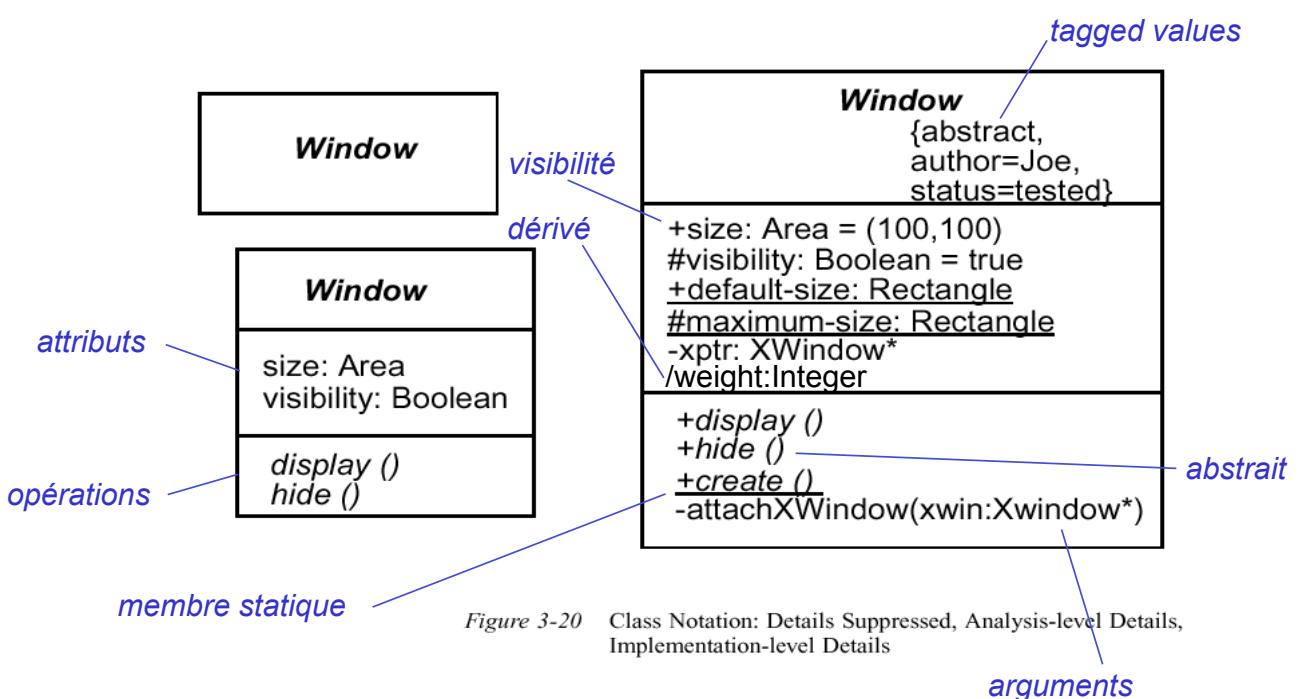


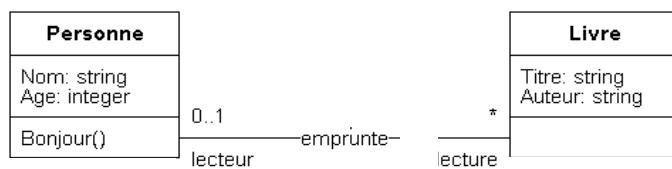
Figure 3-20 Class Notation: Details Suppressed, Analysis-level Details, Implementation-level Details

© 2013 zam@dauphine.fr



Exemple complet (1)

- Classe, association, multiplicité, rôle

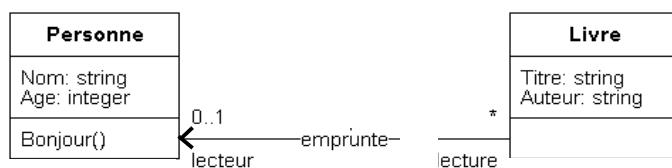


© 2013 zam@dauphine.fr



Exemple complet (1)

- Navigabilité : mono/bi directionnel

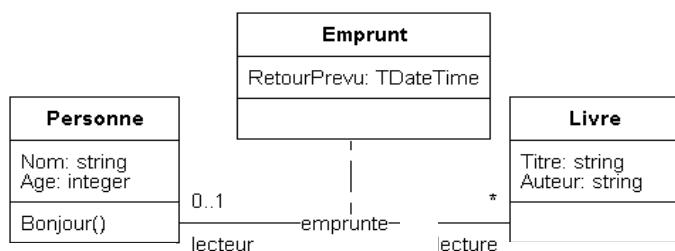


© 2013 zam@dauphine.fr



Exemple complet (2)

- Association + Classe = Classe d'association

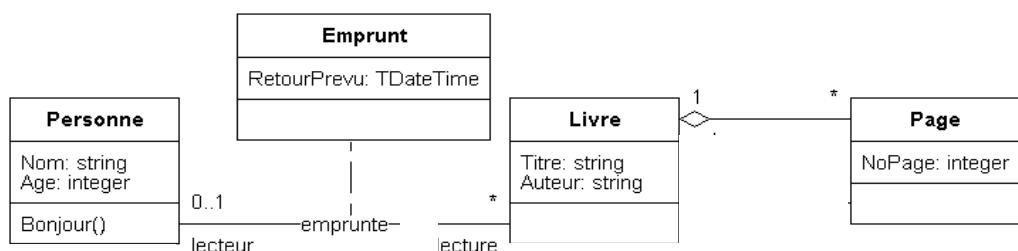


© 2013 zam@dauphine.fr



Exemple complet (3)

- Agrégation = association ensemble-partie

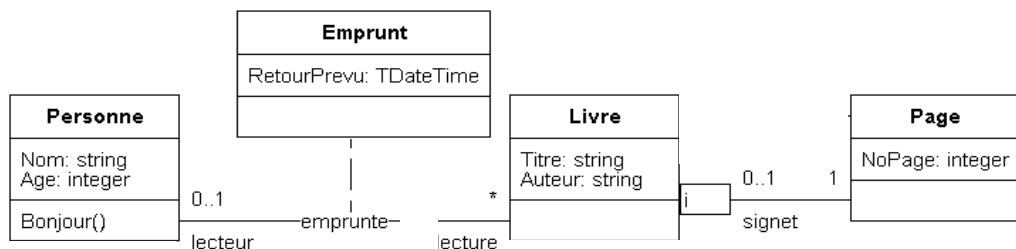


© 2013 zam@dauphine.fr



Exemple complet (4)

- Qualifieur = filtre d'association / partition

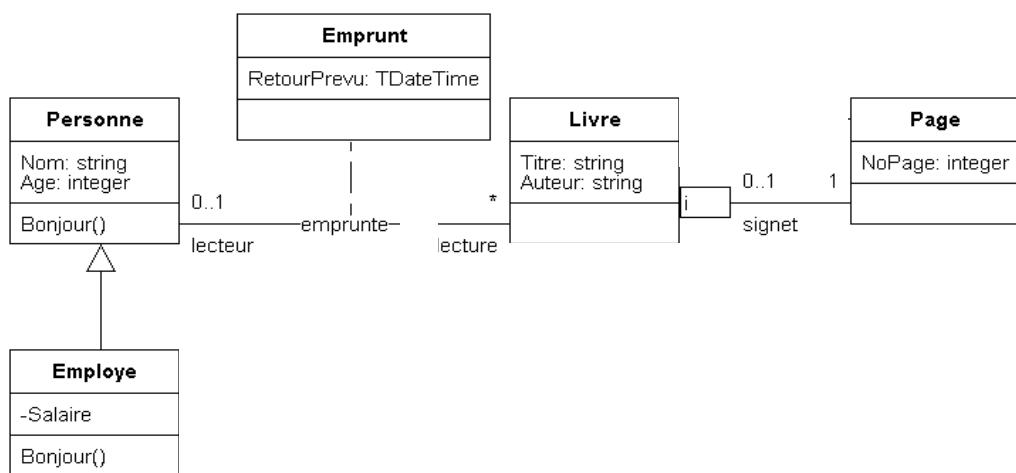


© 2013 zam@dauphine.fr



Exemple complet (5)

- Généralisation

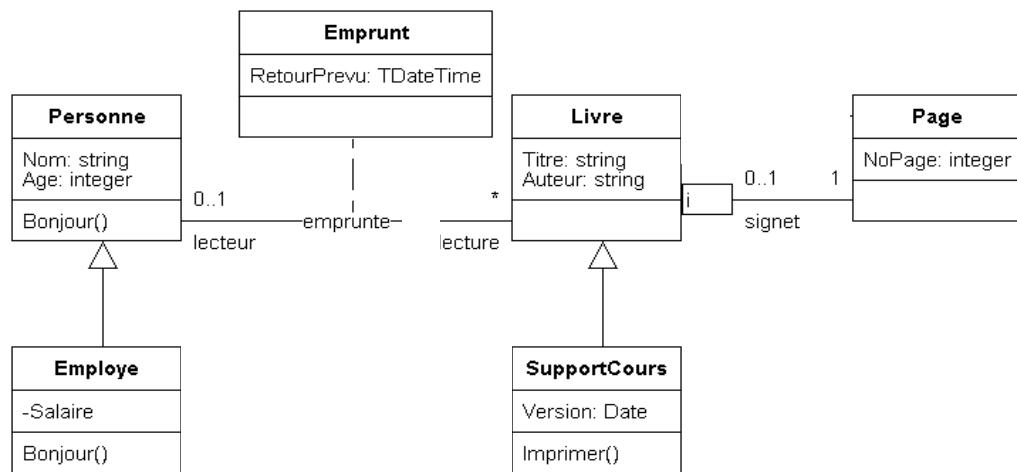


© 2013 zam@dauphine.fr



Exemple complet (6)

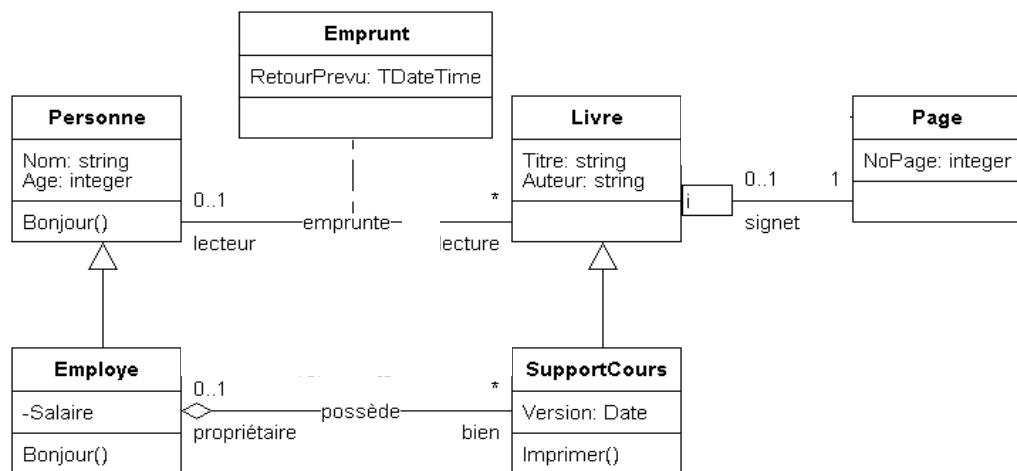
- Généralisation



© 2013 zam@dauphine.fr



Exemple complet (7)

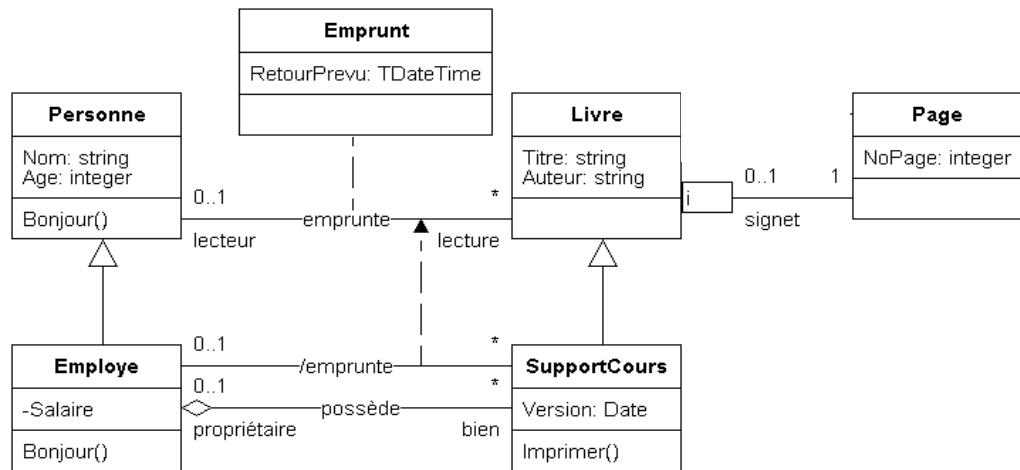


© 2013 zam@dauphine.fr



Exemple complet (8)

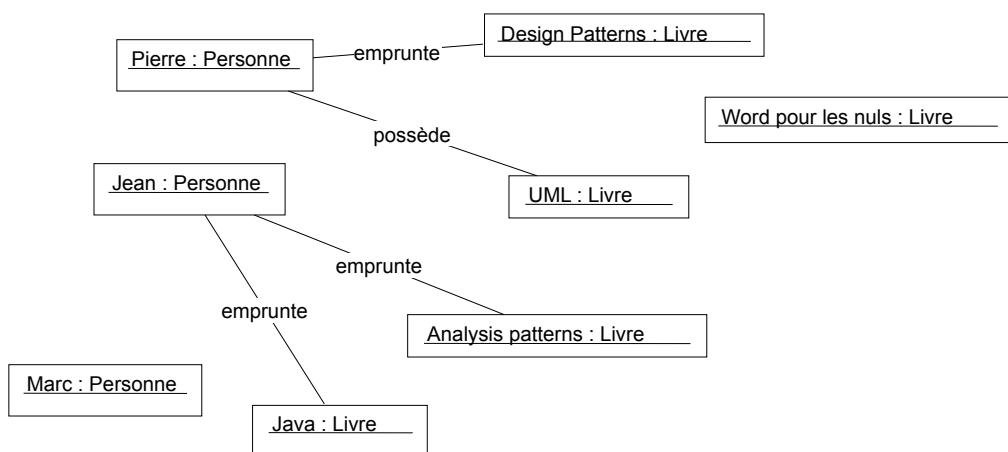
- Association dérivée



© 2013 zam@dauphine.fr

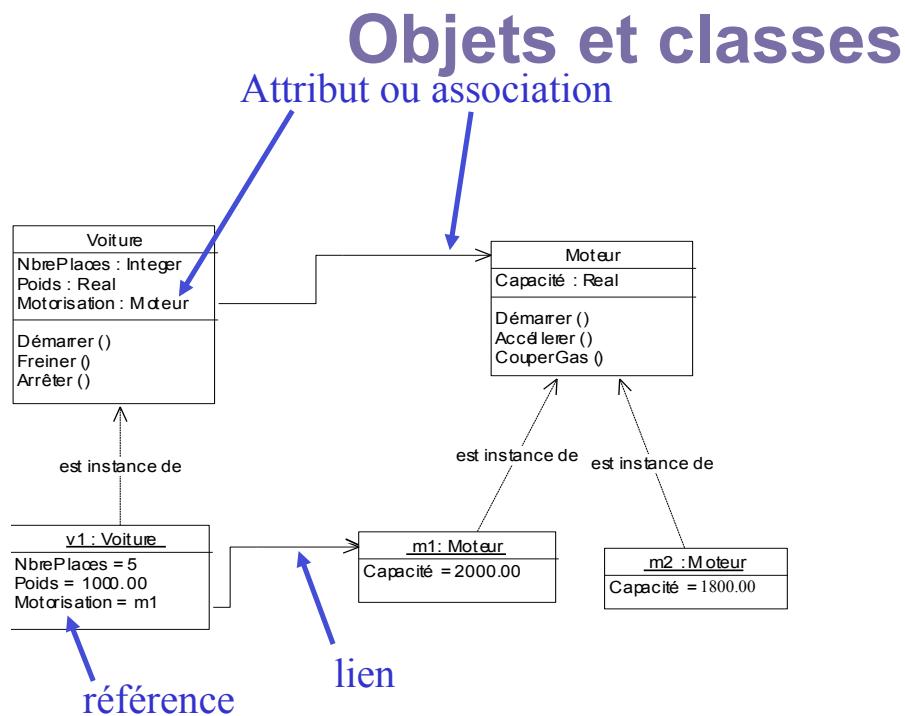


Objets et classes



Exercice : trouvez l'erreur

© 2013 zam@dauphine.fr



© 2013 zam@dauphine.fr



Conseils

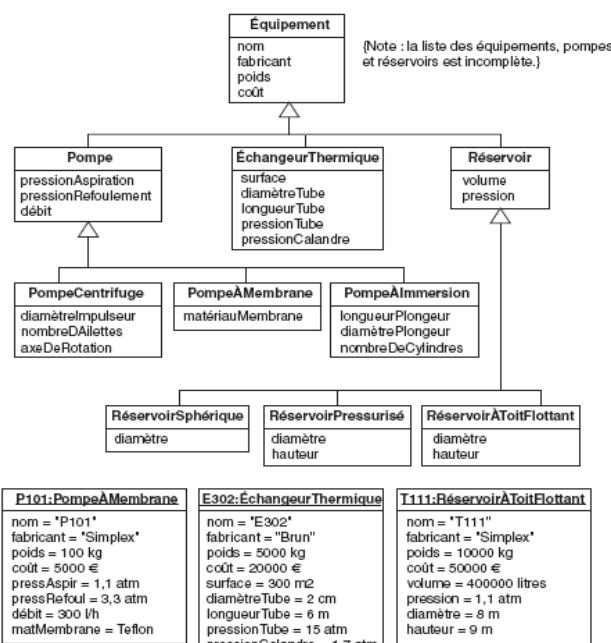
- UML est un moyen de communication :
 - utilisez une notation simple...
 - adaptée à la phase en cours...
 - et multipliez les vues...
 - en choisissant les classes essentielles

© 2013 zam@dauphine.fr



Cahier d'exercices

Diagramme de classe



1. Exercice de lecture

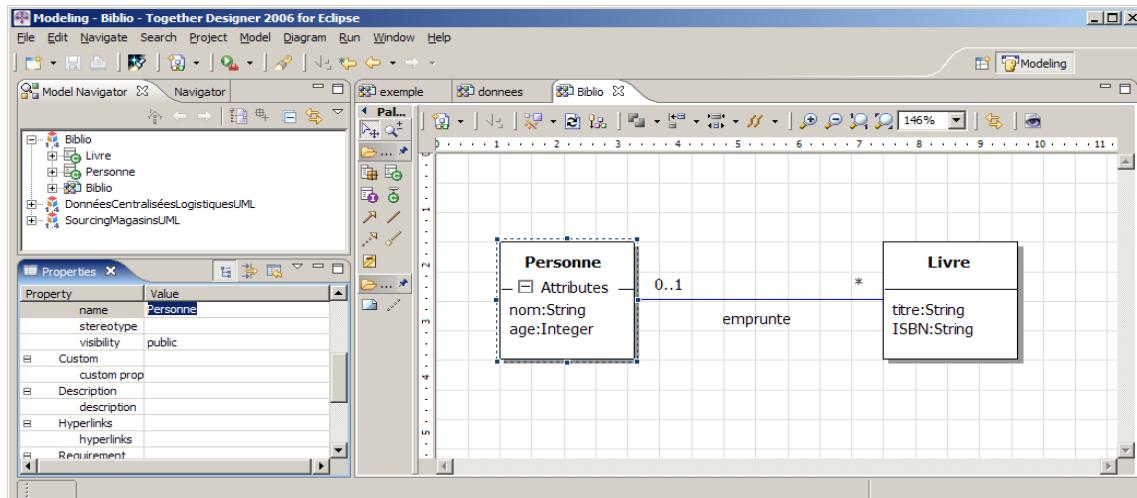
- Identifier les concepts UML
- Interpréter le diagramme
- Vérifier la cohérence entre objets et classes
- Proposer une évolution

Figure 3.24 • Hiérarchie d'héritage multilevel avec instances. La généralisation organise les classes en fonction de leurs similarités et de leurs différences, structurant ainsi la description des objets.



2. Exercice outillé

- Prise en main de l'outil
 - Reprendre un exemple déjà vu
 - Quelle est la différence entre modèle et vue ?



3. Exercices d'abstraction

- 3.1 (3) Tracez un diagramme de classes à partir du diagramme d'objets de la figure E3.1.



Figure E3.1 • Diagramme d'objets pour une partie de l'Europe.

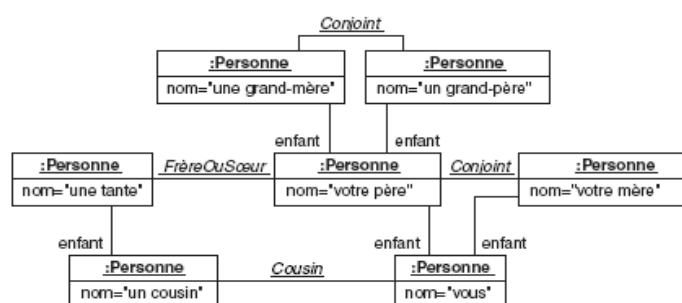


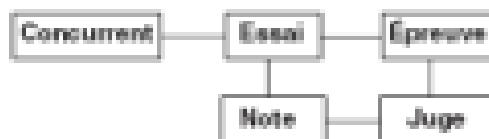
Figure E3.4 • Diagramme d'objets pour une partie de votre arbre généalogique.

4. Exercice de modélisation

- 3.17 (6) La figure E3.9 est un diagramme de classes pouvant être employé pour développer un système destiné à simplifier l'organisation et la notation de compétitions sportives telles que la gymnastique, le plongeon ou le patinage artistique. Plusieurs épreuves existent auxquelles plusieurs concurrents peuvent concourir. Chaque concurrent peut participer à plusieurs épreuves et chaque épreuve réunit plusieurs concurrents.

Chaque épreuve a plusieurs juges qui notent subjectivement les performances des concurrents. Un juge note chaque concurrent lors d'une épreuve. Dans certains cas, un juge peut noter plus d'une épreuve.

Le classement de la compétition s'effectue sur l'ensemble des essais. À chaque essai, le concurrent tente de réaliser la meilleure performance de l'épreuve. Les essais sont notés par un panel de juges pour cette épreuve et un résultat global est calculé. Ajoutez les symboles de multiplicité au diagramme.



© 2013 zam@dauphine.fr

5. Fichiers et répertoires

Dans un système de fichiers, on trouve des fichiers et des répertoires.

On peut faire plusieurs remarques à leur propos :

- Un répertoire peut contenir des fichiers et d'autres répertoires. Par contre, un fichier ou un répertoire ne peut appartenir, au plus, qu'à un répertoire.
- On peut effectuer un certain nombre d'opérations sur les fichiers : les renommer, les déplacer, les supprimer, les éditer, ...
- On peut effectuer un certain nombre d'opérations sur les répertoires : les renommer, les déplacer, les supprimer, les partager, leur ajouter un fichier ou un répertoire, leur retirer un fichier ou un répertoire, ...

L'exercice consiste à réaliser un diagramme de classes qui modélise ce problème.

© 2013 zam@dauphine.fr



6. Figures géométriques

- Considérons un logiciel de dessin vectoriel. Dans celui-ci, il est possible de tracer et de manipuler diverses formes géométriques : points, segments de droites, carrés, rectangles, cercles et ellipses. Quelque soit le type de la forme, il est possible de réaliser un certain nombre d'opérations sur celle-ci même si chaque forme a ses caractéristiques propres (le rayon pour un cercle, la longueur et la largeur pour un rectangle, ...). Le logiciel peut ainsi l'afficher, la déplacer, ...
- Représentez dans un diagramme de classes les différentes formes et les relations qui les lient afin de satisfaire à ces exigences

© 2013 zam@dauphine.fr



Autres exercices

- Modélisation légère, non-dirigée
 - Une ruche, une abeille
 - Un véhicule, un avion, un ordinateur, un bateau
 - Une maison (menuiserie comprise)
- Métamodélisation
 - Modéliser les concepts d'un diagramme de classes

© 2013 zam@dauphine.fr



Diagramme d'états

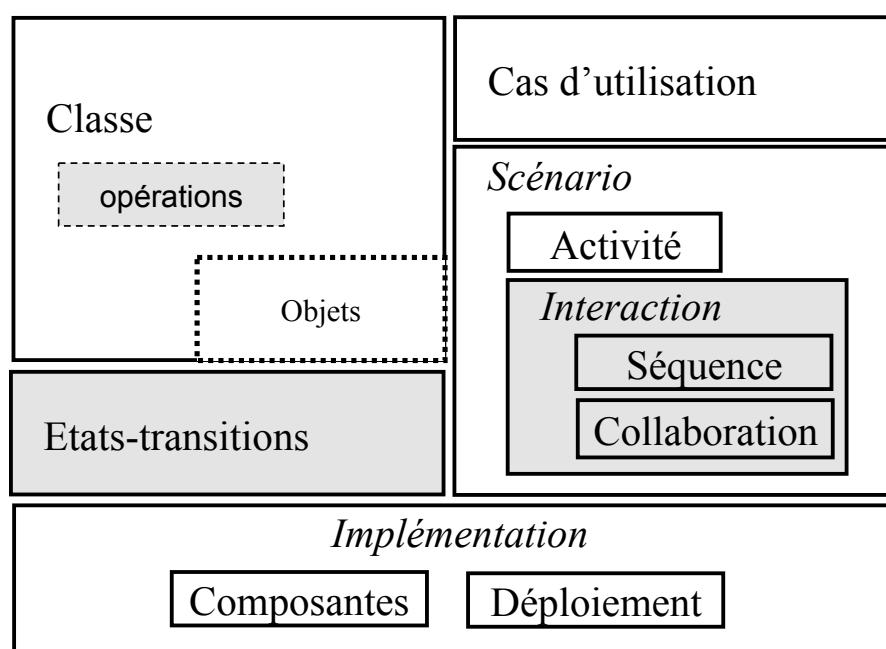
ou « de transitions d'états »

ou « d'état-transitions »

ou « machines à états (statecharts) »



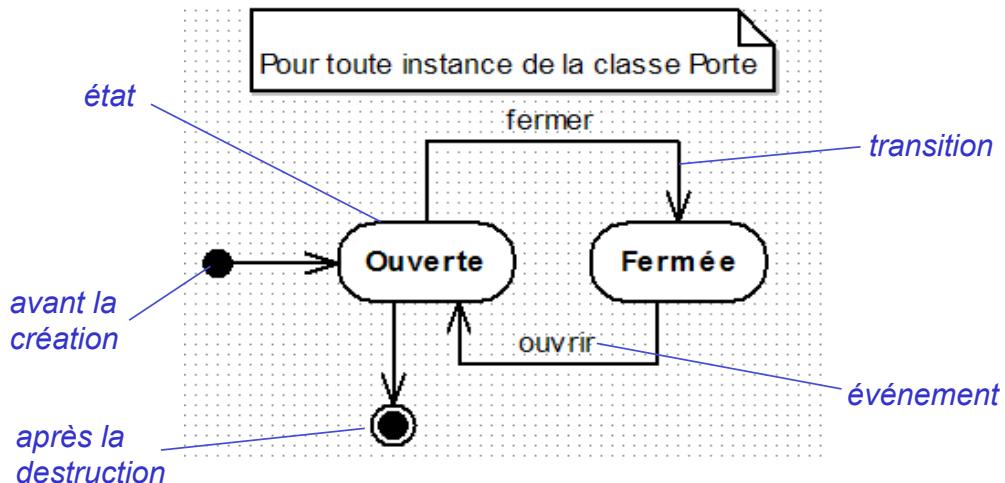
Diagrammes dynamiques / objets





Objectif

- Décrire les états remarquables d'un **seul** objet et les transitions possibles



© 2013 zam@dauphine.fr



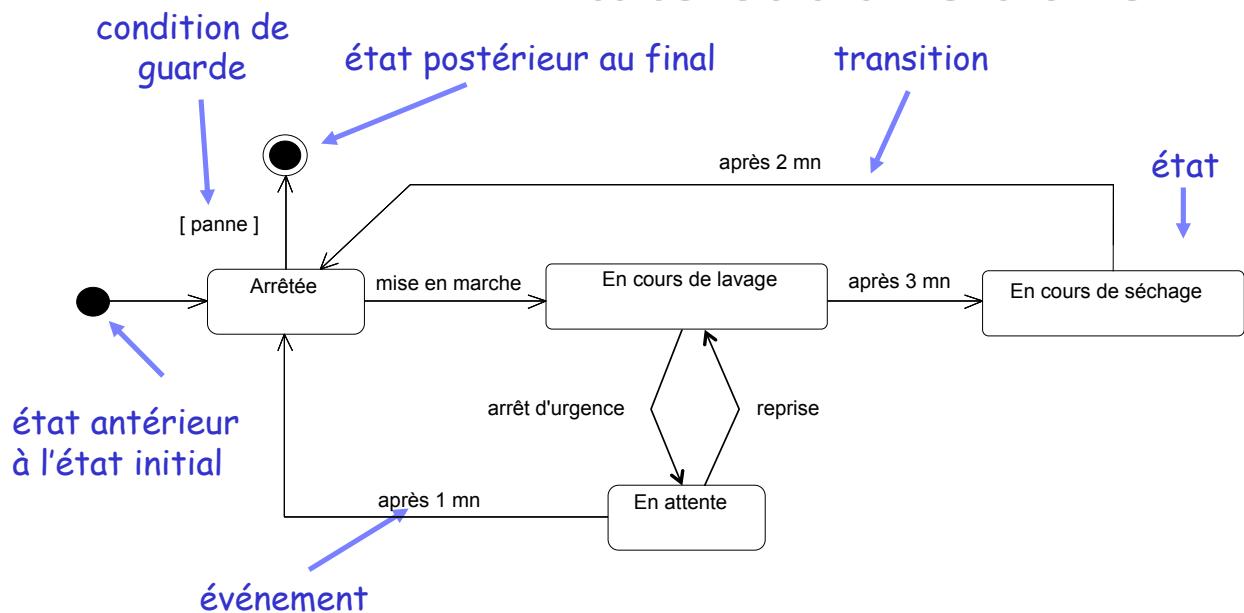
Définitions

- « Etat » = valeur remarquable d'un objet
- « Transition » = changement d'état d'un objet, provoquée par un « événement », si « condition » respectée
- « Événement » = assimilable à appel d'opération
- Etats prédéfinis
 - « Avant la création »
 - « Après la destruction »

© 2013 zam@dauphine.fr



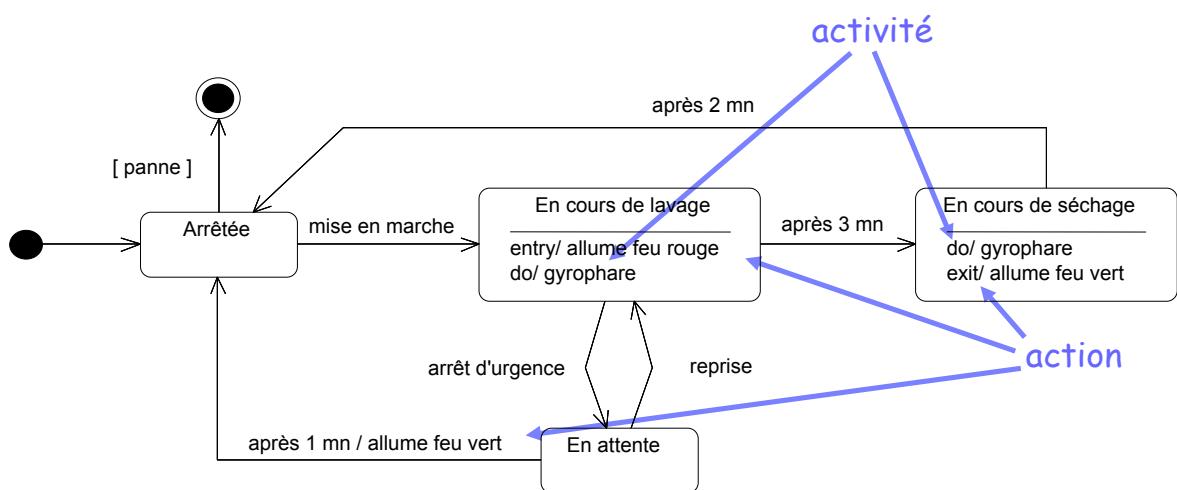
États et transitions



© 2013 zam@dauphine.fr



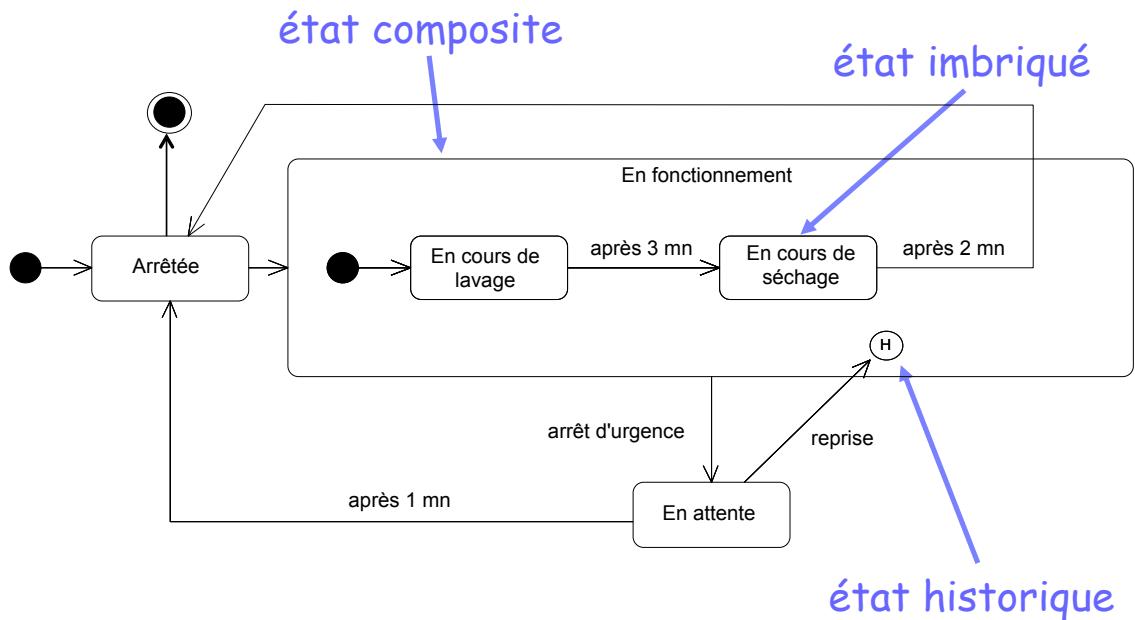
Actions et activités



© 2013 zam@dauphine.fr



États imbriqués

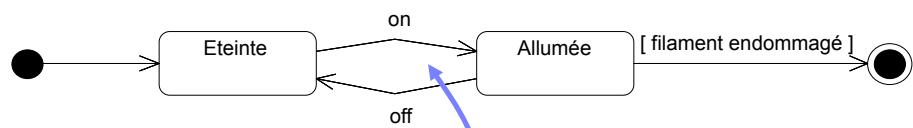


© 2013 zam@dauphine.fr

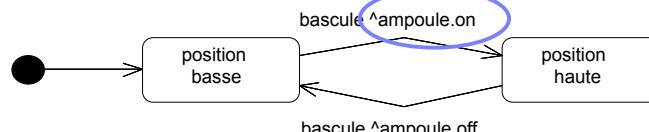


Communication entre objets

Ampoule



Interrupteur



© 2013 zam@dauphine.fr

- Transitions, événements (avec paramètres), actions (entry, exit), conditions

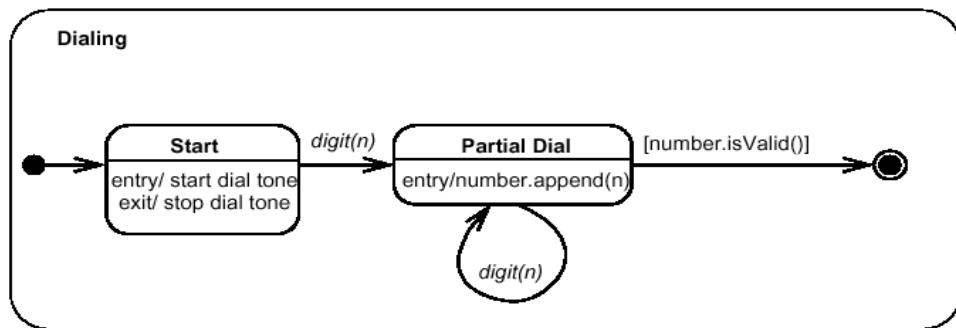


Figure 3-73 Sequential Substates

© 2013 zam@dauphine.fr

- Actions prédéfinies et personnalisées
- (Transitions internes)

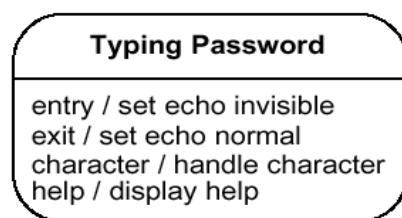
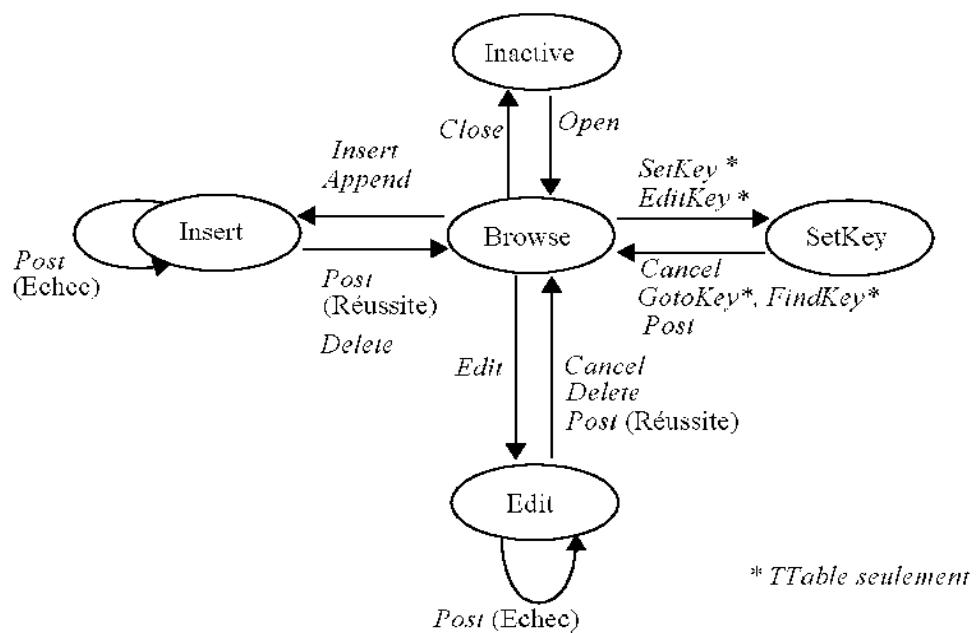


Figure 3-72 State

© 2013 zam@dauphine.fr

© 2013 zam@dauphine.fr

TDataSet

© 2013 zam@dauphine.fr



Conseils

- États d'un **seul** objet
 - Vue égoïste
- Choisir les classes adaptées
 - États remarquables qui conditionnent le comportement de l'objet : applicabilité des méthodes
- États mutuellement exclusifs
 - Éviter les états parallèles
- Déterminisme
 - état départ + événement + condition → état d'arrivée unique
- Attention aux objets complexes

© 2013 zam@dauphine.fr



Cahier d'exercices

Diagramme d'états

1. Exercice de lecture

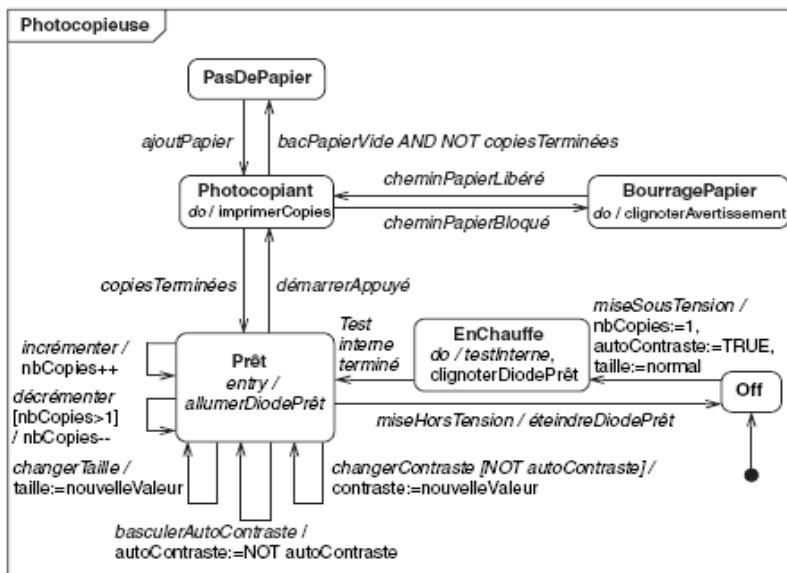


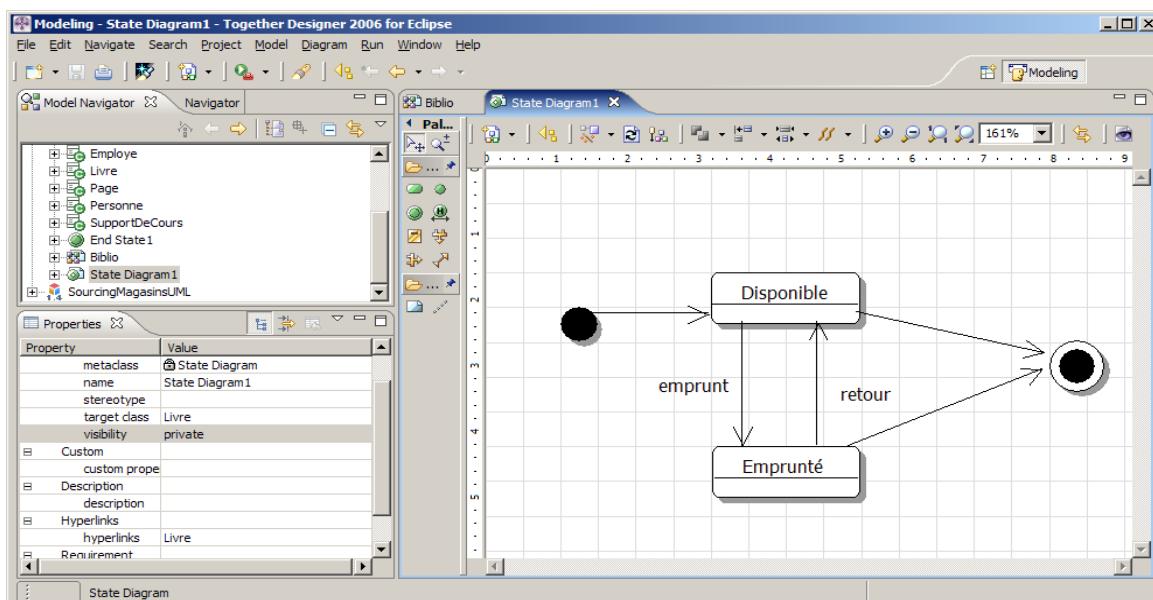
Figure E5.4 • Diagramme d'états d'une photocopieuse.

- Identifier les concepts UML
- Interpréter le diagramme
- Proposer un diagramme de classes associé
- Proposer une évolution

© 2013 zam@dauphine.fr

2. Exercice outillé

- Prise en main de Outil



© 2013 zam@dauphine.fr



3. Feu tricolore

- Un feu tricolore passe successivement du vert au orange, du orange au rouge et du rouge au vert à intervalles de temps réguliers. Il est possible de faire passer le feu tricolore du fonctionnement normal à l'orange clignotant en cas de besoin. Lorsqu'il quitte ce mode, le feu passe automatiquement au vert.
- L'exercice consiste à modéliser le feu tricolore à l'aide d'un diagramme d'états.

© 2013 zam@dauphine.fr



4. Montre digitale

- Une montre digitale simple possède un cadran et deux boutons de réglage, nommés *A* et *B*. Elle a deux modes de fonctionnement : affichage de l'heure et réglage de l'heure. En mode affichage, elle affiche l'heure et les minutes, séparées par le caractère deux-points qui clignote.
- Le mode de réglage a deux sous-modes : réglage de l'heure et réglage des minutes. Le bouton *A* sélectionne le mode. Chaque fois que l'on appuie dessus, le mode avance selon la séquence suivante : affichage, réglage de l'heure, réglage des minutes, affichage, etc. Dans un sous-mode, le bouton *B* avance les heures ou les minutes chaque fois que l'on appuie dessus. Les boutons doivent être relâchés avant de pouvoir générer un autre événement.
- Tracez le diagramme d'états correspondant.

© 2013 zam@dauphine.fr

5. Commandes d'un véhicule

- Il est demandé de modéliser à l'aide du formalisme du diagramme d'états-transitions les commandes d'un véhicule. Plus particulièrement, nous souhaiterions modéliser par des états, des transitions et événements le fait de démarrer, d'accélérer, de freiner, de changer de vitesse, etc.

© 2013 zam@dauphine.fr

6. Autres exercices

- Avion
- Réveil
- Lave linge
- La porte du frigidaire
- Métamodélisation

© 2013 zam@dauphine.fr

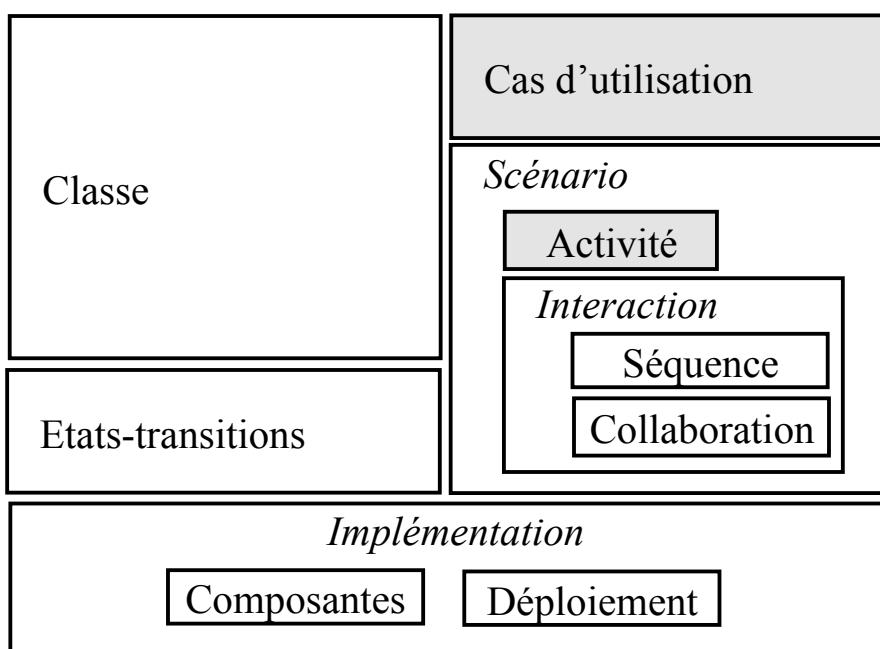


Diagramme de cas d'utilisation

(use case)



Diagrammes UML





Objectif

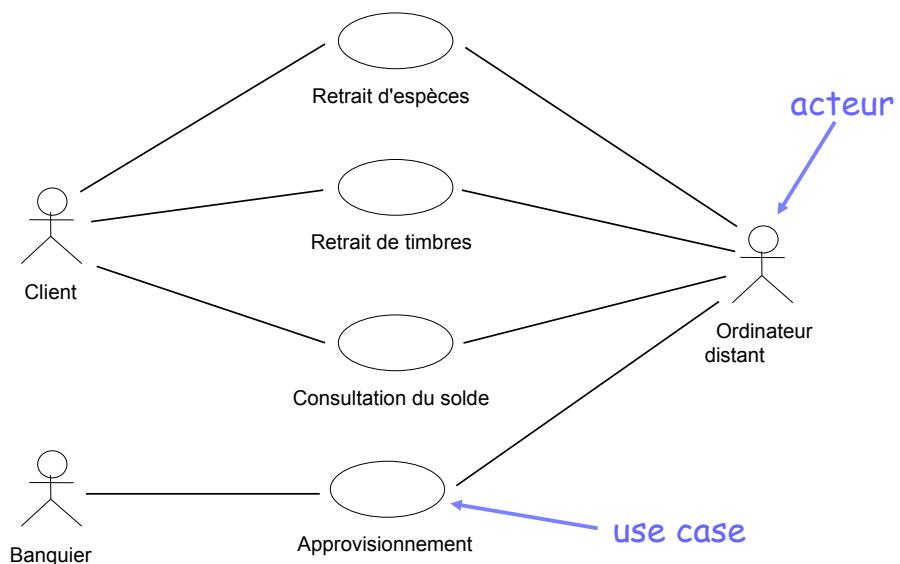
- ... du *use case*
 - Capturer les besoins du client sous la forme d'une interaction entre l'utilisateur et le système

- ... du diagramme de *use cases*
 - Mettre en évidence les acteurs qui participent aux *use cases* et montrer les relations entre ces derniers.

© 2013 zam@dauphine.fr



Acteurs et use cases



© 2013 zam@dauphine.fr



Description d'un use case

Nom retrait d'espèces

Description le but de ce use case est pour le client de retirer des espèces

Scénario principal

1. le client introduit sa carte
2. le distributeur vérifie la carte
3. le client saisit son code secret
4. le distributeur vérifie le code
5. le client choisit l'opération « retrait d'espèces »
6. le client spécifie la somme à retirer
7. le distributeur débite le compte
8. le distributeur rend la carte
9. le client prend la carte
10. le distributeur fournit les billets
11. le client prend les billets



Description d'un use case

Alternative : code incorrect

(à l'étape 4 du scénario principal, le code est incorrect)

1. le distributeur demande à nouveau le code secret
2. le client saisit son code
3. le distributeur vérifie le code

(si le code est correct, on retourne à l'étape 5 du scénario principal sinon, s'il s'agit de la deuxième tentative on rejoue l'étape 1 de l'alternative ; si il s'agit du troisième échec on passe à l'étape 4 de l'alternative)

4. le distributeur saisit la carte bancaire et avertit le client

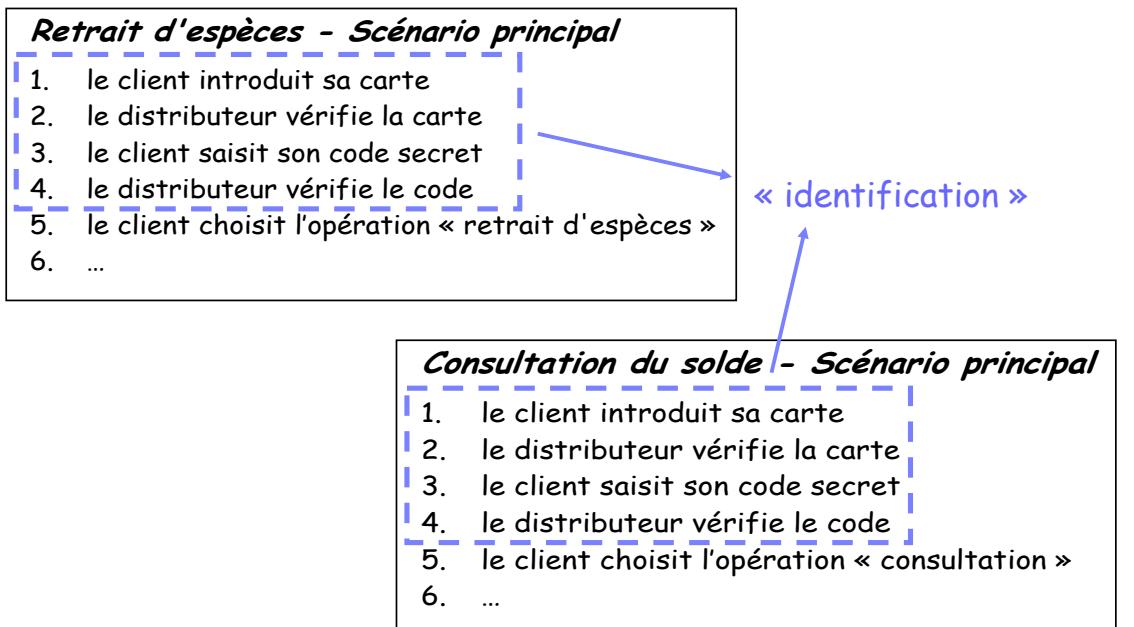
Pré-condition le client doit disposer d'une carte bancaire

Post-condition de succès le client dispose de sa carte et de l'argent en espèces

Post-condition d'échec le client s'est vu saisir sa carte bancaire



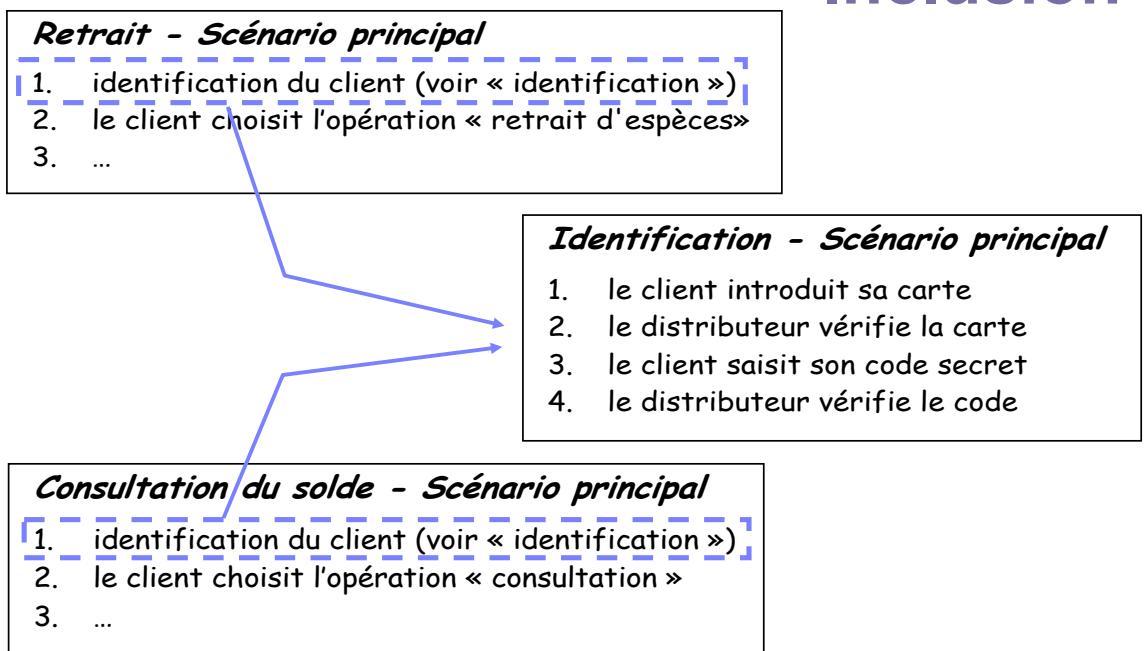
Inclusion



© 2013 zam@dauphine.fr



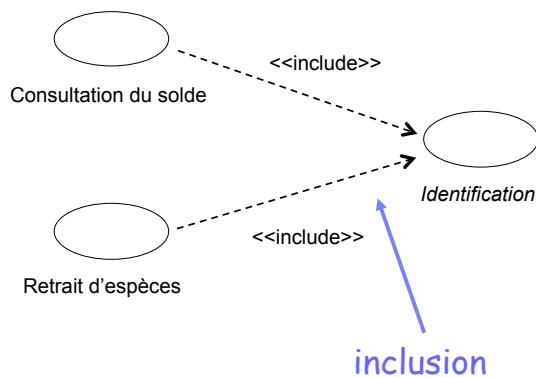
Inclusion



© 2013 zam@dauphine.fr



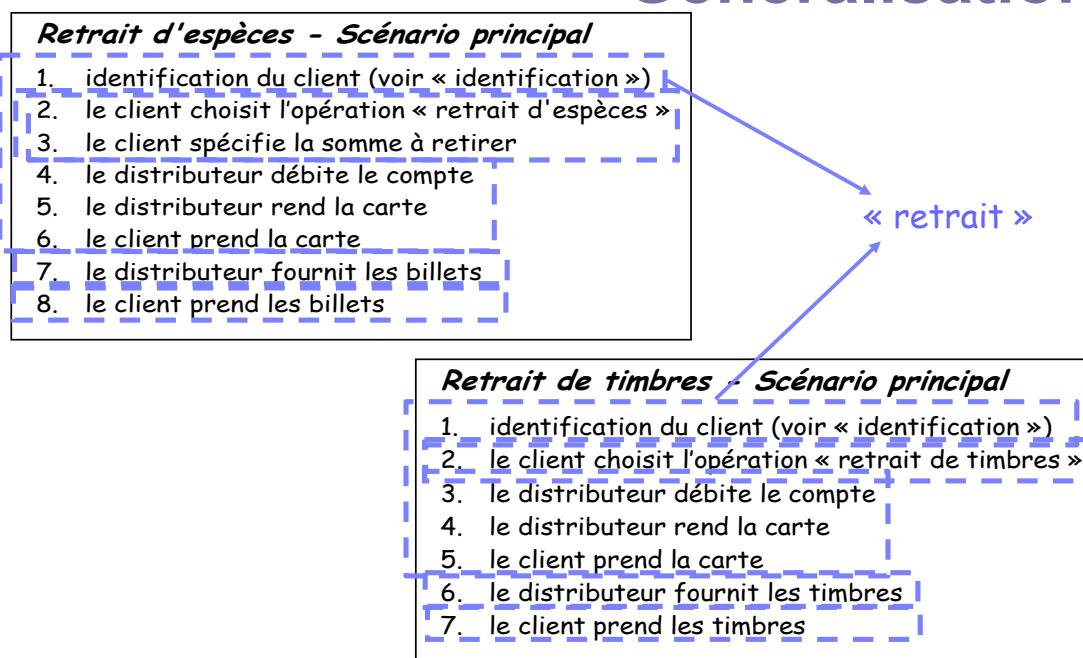
Inclusion



© 2013 zam@dauphine.fr



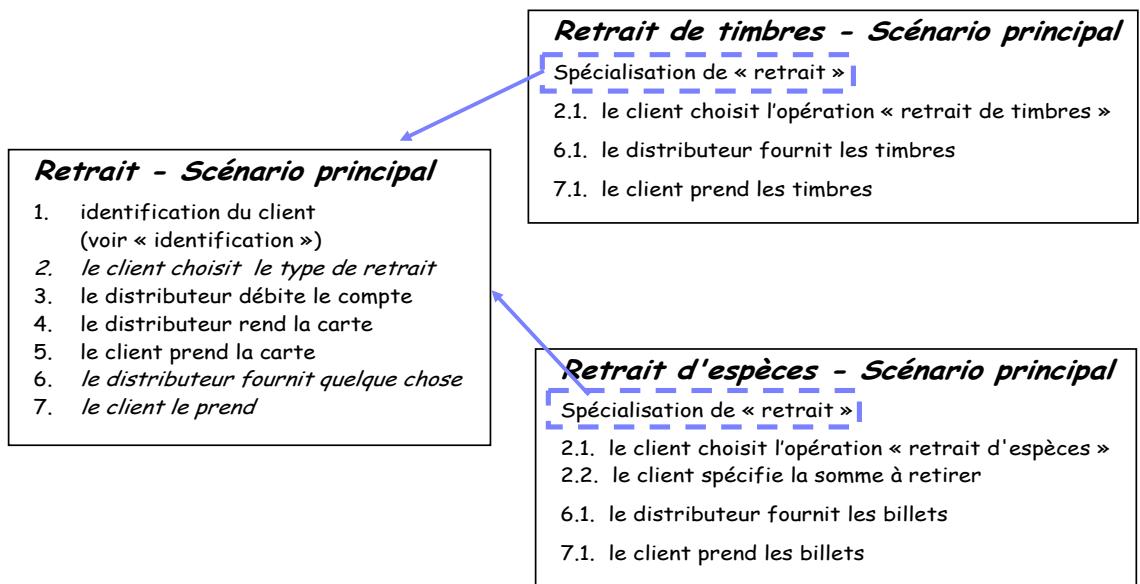
Généralisation



© 2013 zam@dauphine.fr



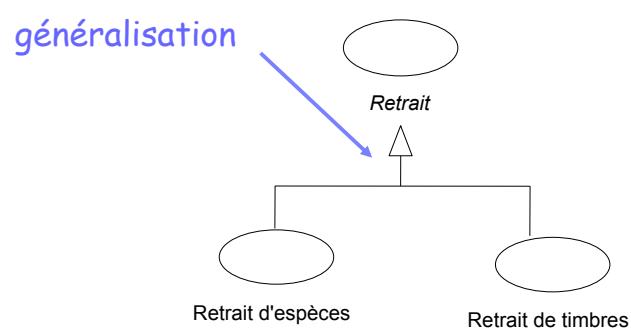
Généralisation



© 2013 zam@dauphine.fr



Généralisation



© 2013 zam@dauphine.fr



Extension

Retrait d'espèces - Scénario principal

2. ...
3. le client spécifie la somme à retirer
4. le distributeur débite le compte
5. le distributeur rend la carte
6. le client prend la carte
7. le distributeur fournit les billets
8. le client prend les billets

Retrait d'espèces avec reçu - Scénario principal

2. ...
3. le client spécifie la somme à retirer
4. le distributeur débite le compte
5. le client demande un reçu
6. le distributeur rend la carte
7. le client prend la carte
8. le distributeur fournit les billets
9. le client prend les billets
10. le distributeur fournit le reçu
11. le client prend le reçu

© 2013 zam@dauphine.fr



Extension

Retrait d'espèces - Scénario principal

2. ...
 3. le client spécifie la somme à retirer
 4. le distributeur débite le compte
- Point d'extension A*
5. le distributeur rend la carte
 6. le client prend la carte
 7. le distributeur fournit les billets
 8. le client prend les billets
- Point d'extension B*

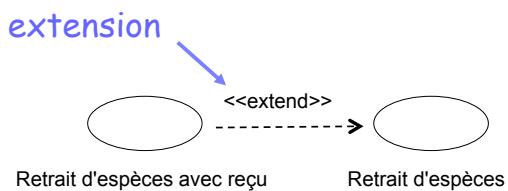
Retrait d'espèces avec reçu - Scénario principal

- Étend « retrait d'espèces »
- A.1. le client demande un reçu
 - B.1. le distributeur fournit le reçu
 - B.2. le client prend le reçu

© 2013 zam@dauphine.fr



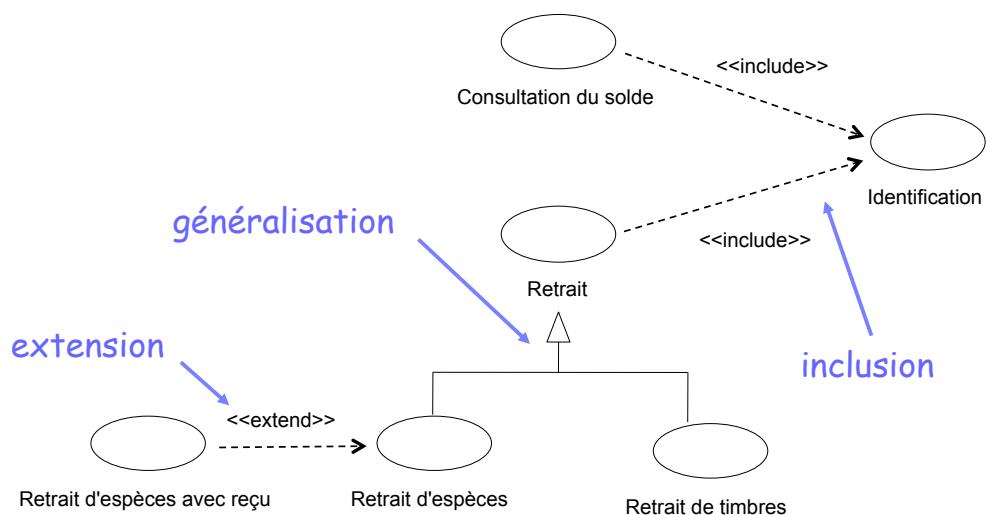
Extension



© 2013 zam@dauphine.fr



Relations entre use cases



© 2013 zam@dauphine.fr



Conseils

- se concentrent sur le fonctionnel interactif
- sont largement utilisés ; Club UML : 64%
- ont une granularité variable
- comment identifier les « Use Case » ?
 - critère principal = séquence d'actions/événements, décrite sous forme :
 - textuelle (au début)
 - graphique (par la suite, à l'aide d'autres diagrammes, dits de scénario)

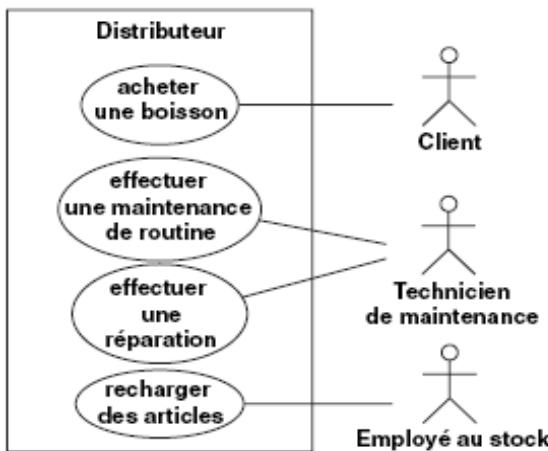
© 2013 zam@dauphine.fr



Cahier d'exercices

Diagramme de cas d'utilisation

1. Le distributeur de boissons



- Interpréter le diagramme
- Prise en main de l'outil
- Proposer un résumé pour chaque cas d'utilisation (p. 165)

Figure 7.3 • Diagramme du cas d'utilisation d'un distributeur automatique. Un système implique un ensemble de cas d'utilisation et un ensemble d'acteurs.

© 2013 zam@dauphine.fr

Complétez les sections suivantes

- **Cas d'utilisation :** Acheter une boisson
- **Résumé :**
- **Acteurs :**
- **Scénario :** <dans une note>
- **Préconditions :**
- **Exceptions :**
- **Postconditions :**

© 2013 zam@dauphine.fr



2. Le garagiste

- L'activité de monsieur Dupond, garagiste, consiste essentiellement à assurer le contrôle technique et la réparation des véhicules ainsi que la vente de pièces détachées. Le garage dispose de plusieurs services :
 - Le service technique qui s'occupe de l'entretien, de la réparation et du contrôle technique des véhicules.
 - Le service commercial qui assure l'accueil des clients, réalise les ventes et édite factures, devis et comptes-rendus de contrôle technique.
 - Le service logistique qui gère notamment les stocks.
- Dans le cadre de son expansion, monsieur Dupond souhaiterait disposer d'un logiciel lui permettant de gérer stocks, factures, devis et comptes-rendus de contrôle technique ainsi que de gérer la liste de clients.
- L'exercice consiste à exprimer ces besoins sous la forme d'un diagramme de cas d'utilisation. On prendra soin de détailler textuellement l'un des cas d'utilisation au choix.

© 2013 zam@dauphine.fr



2. Autres exercices

- Bibliothèque
- Explorateur Windows
- Gestion commerciale
- Pompe à essence [Blaha, p193]

© 2013 zam@dauphine.fr



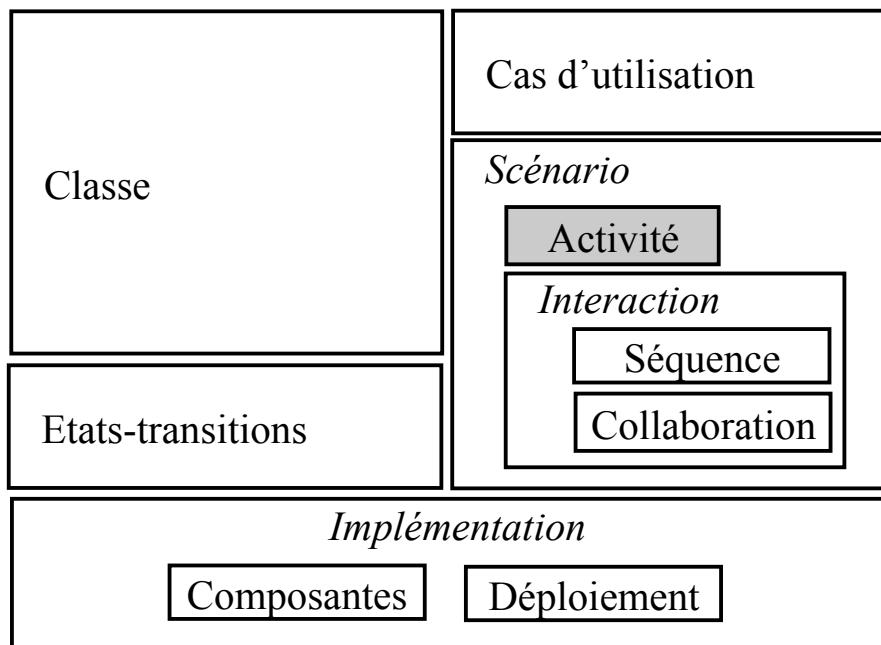
Suites

- Le diagramme de cas d'utilisation introduit les interactions utilisateur-système
- Pour expliciter les scénarios des « Use Cases », on utilise des diagrammes dynamiques spécialisés dits "de scénario"
 - **Diagramme d'activité**
 - Diagramme d'interaction

© 2013 zam@dauphine.fr



Diagramme d'activités



© 2013 zam@dauphine.fr



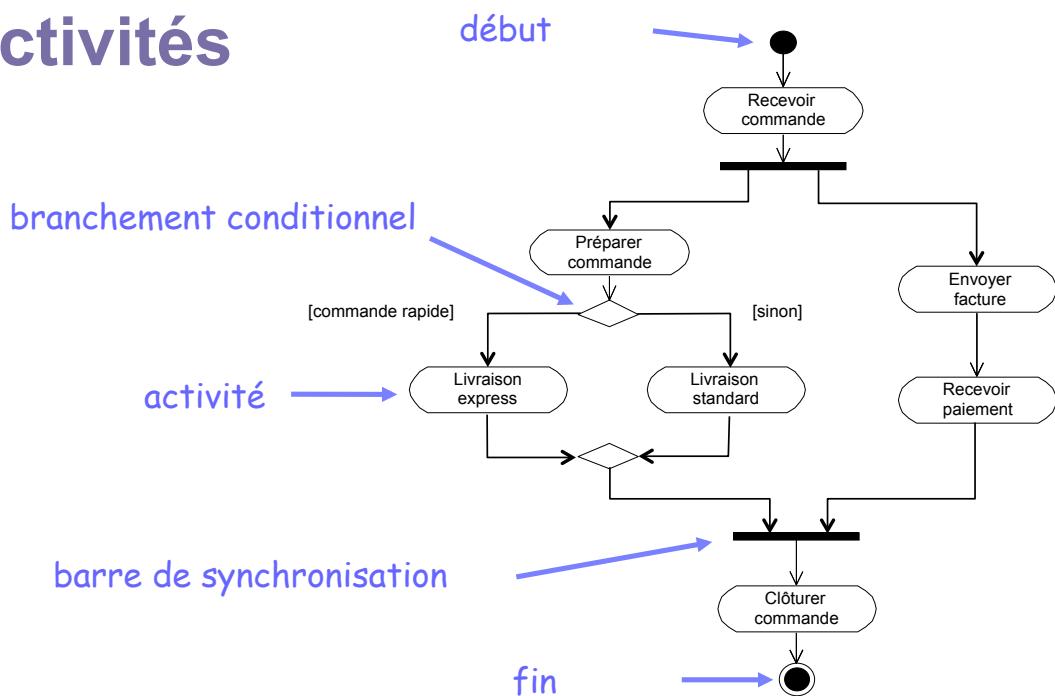
Objectif

- Mettre en évidence les activités d'un processus, d'une méthode, d'un *use case* et montrer leur enchaînement.
- Indiquer les activités exécutables en parallèle.

© 2013 zam@dauphine.fr



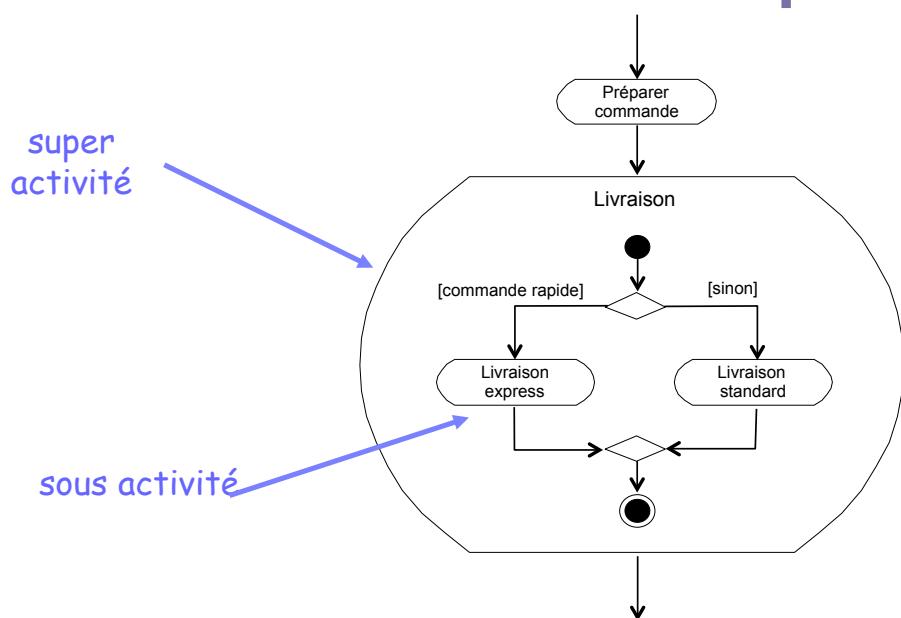
Activités



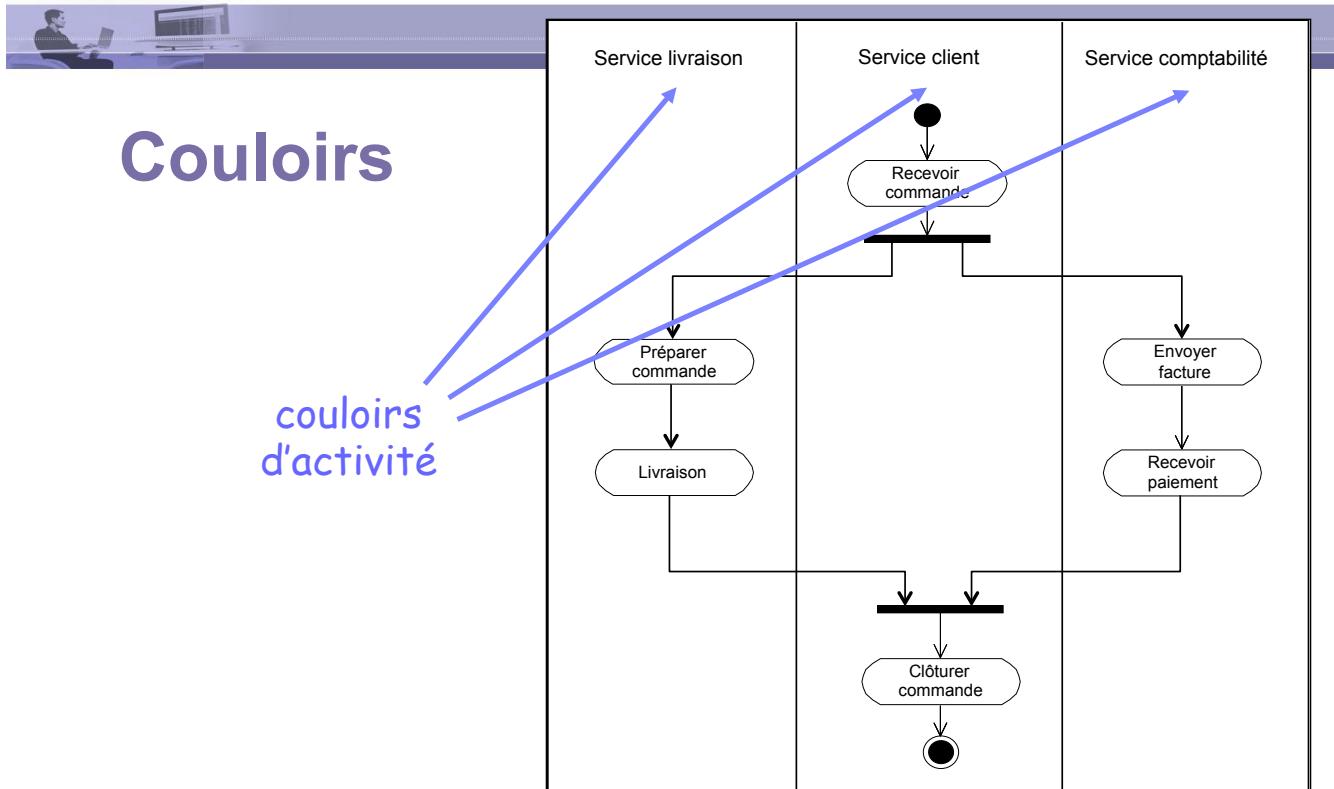
© 2013 zam@dauphine.fr



Activités imbriquées



© 2013 zam@dauphine.fr



© 2013 zam@dauphine.fr

[Blaha&Rumbaugh05] – p192

Le flux d'objets

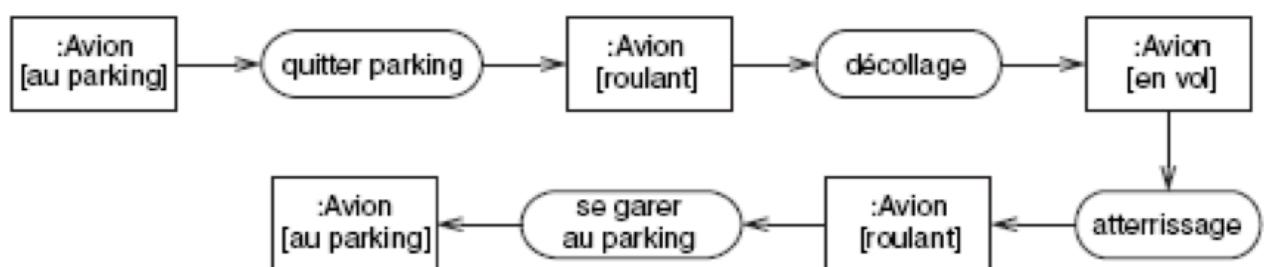


Figure 8.9 • Diagramme d'activités avec flux d'objets. Un diagramme d'activités peut représenter les objets impliqués en entrée ou en sortie d'activités.

© 2013 zam@dauphine.fr



Conseils

Diagramme d'activités

- encourage l'identification des traitements parallélisables (threads)
- description de méthodes complexes
- description de cas d'utilisation (multiples et complexes)
- comportement pour plusieurs classes
- ne pas utiliser pour décrire :
 - la façon dont les objets collaborent
 - le comportement d'un objet au long de son cycle de vie

© 2013 zam@dauphine.fr

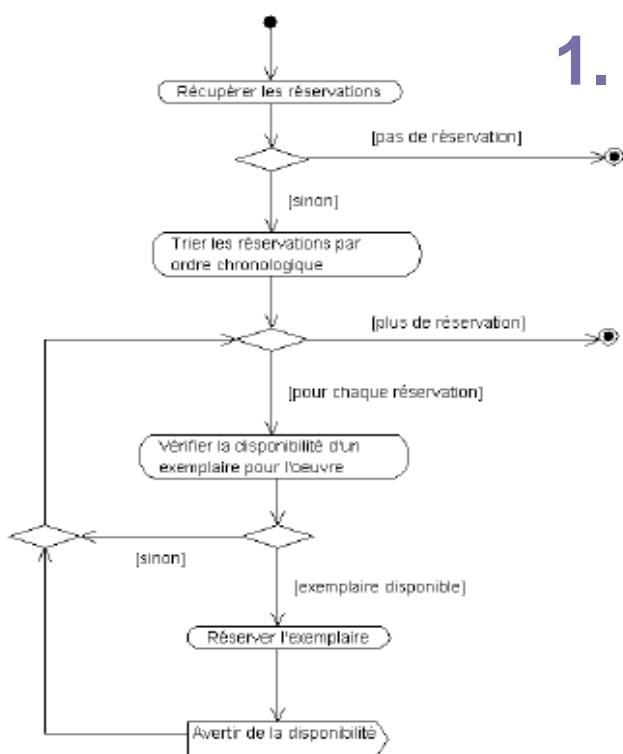


Cahier d'exercices

Diagramme d'activités



1. Exercice de lecture



- Identifier les concepts UML
- Interpréter le diagramme
- Prise en main de l'outil
- Proposer une évolution

© 2013 zam@dauphine.fr



2. Distributeur de boissons

- Tracer un diagramme d'activités selon le scénario du cas d'utilisation « acheter une boisson »
- Dans un premier temps, se contenter des flux de contrôle

© 2013 zam@dauphine.fr



3. Mousse au chocolat

Ingrédients (pour 4 personnes)

- 4 oeufs
- 200g de chocolat
- 100g de beurre
- 4 c à soupe de sucre
- 1 c à café de café fort ou un autre parfum (*optionnel*)

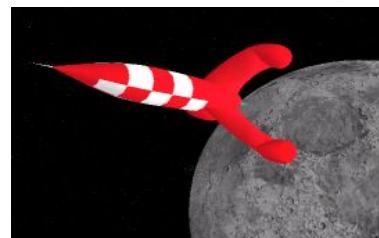
Préparation

1. Battre les jaunes d'oeuf avec le sucre jusqu'à ce qu'ils blanchissent.
2. Faire ramollir le chocolat dans une casserole au bain-marie. L'ajouter ensuite à la préparation, bien remuer, adjoindre le beurre par petites quantités jusqu'à ce que le mélange soit bien lisse.
3. Parfumer la crème avec le café ou le parfum au choix (*optionnel*).
4. Battre les blancs d'oeuf en neige très ferme et les incorporer délicatement à la crème.
5. Laisser reposer 2 heures au frais avant de servir.

L'exercice consiste à montrer — à l'aide d'un diagramme d'activité — comment Marc et Sophie peuvent réaliser cette recette. On pourra aussi faire apparaître les ingrédients et les produits intermédiaires.

© OM

© 2013 zam@dauphine.fr





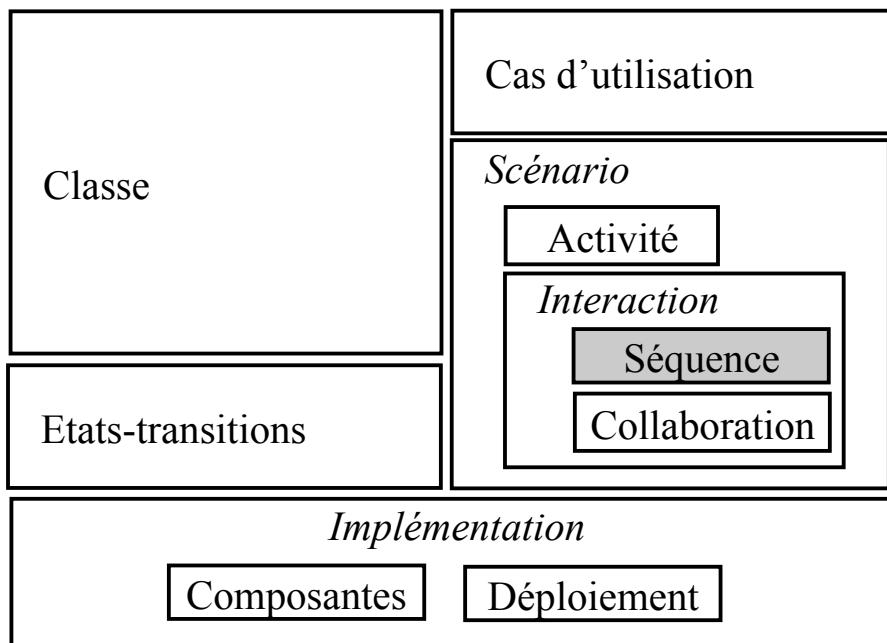
III. Survol des diagrammes restants

Séquence & collaboration
Composantes & déploiement



Diagramme de séquence

Diagrammes UML



© 2013 zam@dauphine.fr

Objectif

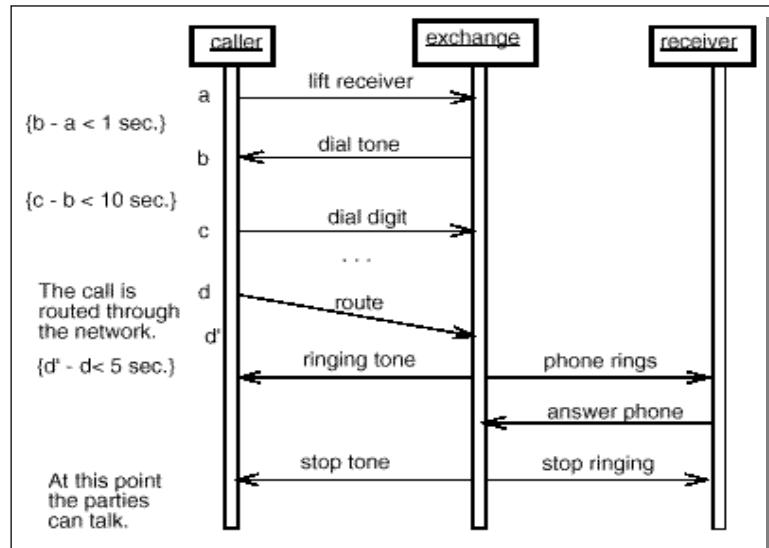
- Représenter les interactions entre les objets de manière séquentielle.

© 2013 zam@dauphine.fr



La notation UML

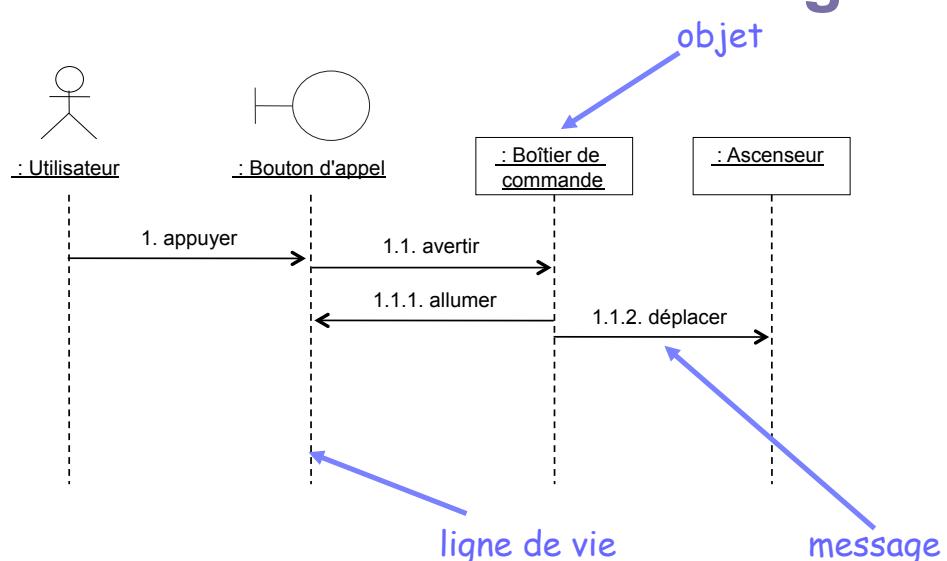
- Diagrammes de séquence



© 2013 zam@dauphine.fr



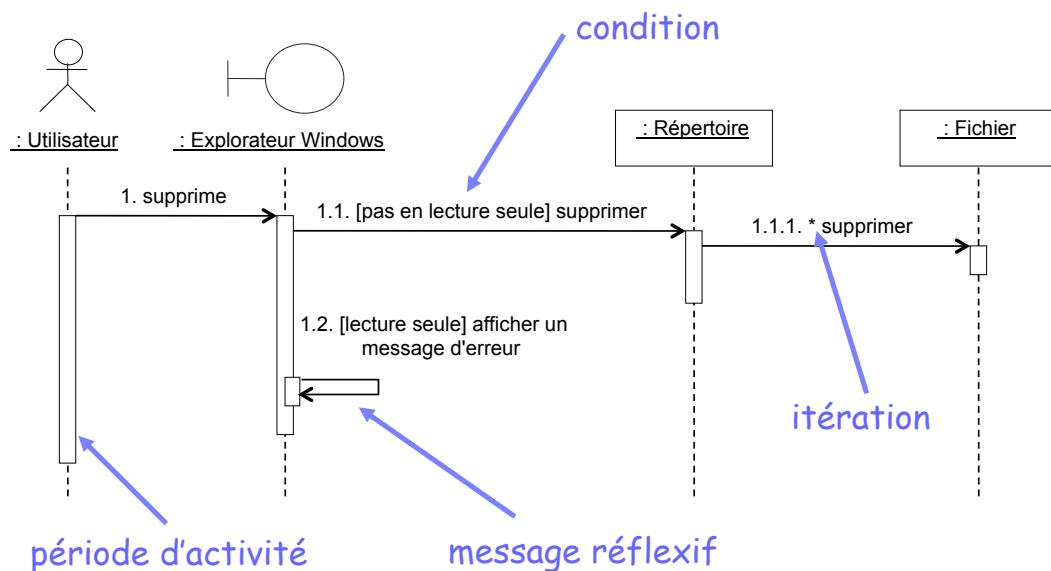
Messages



© 2013 zam@dauphine.fr



Structures de contrôle



© 2013 zam@dauphine.fr



Diagramme de séquence

- Concepts
 - Message synchrone et asynchrone (pam 150)
 - Délai de propagation
 - Création et destruction d'objet
 - Retour implicite et explicite
 - Contrôle centralisé ou décentralisé (pam 154)
 - Contraintes temporelles

© 2013 zam@dauphine.fr



Conseils

- Peut être utilisé
 - en analyse pour décrire un scénario
(mettre le texte en note sur la gauche)
 - en conception pour décrire les appels de méthodes

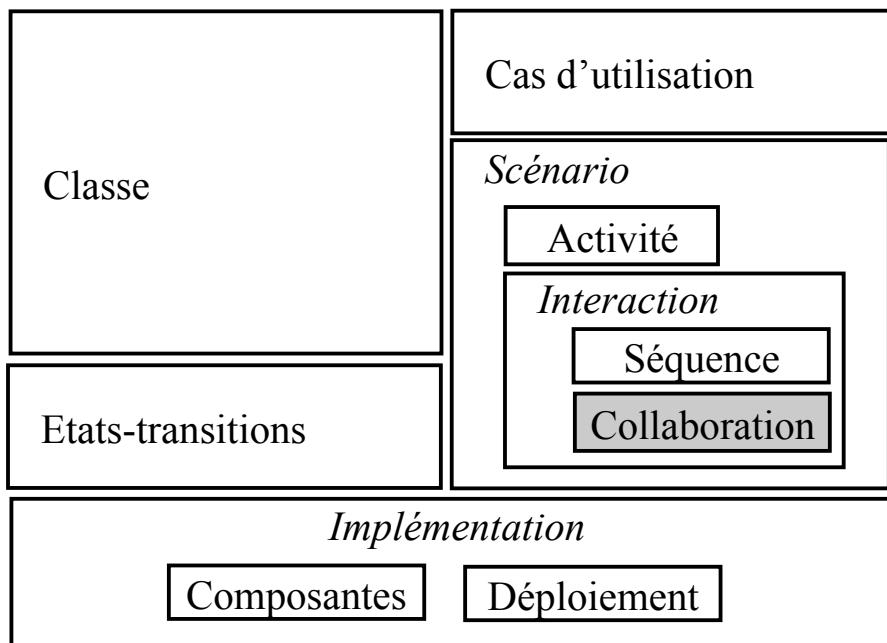
© 2013 zam@dauphine.fr



Diagramme de collaboration



Diagrammes UML



© 2013 zam@dauphine.fr



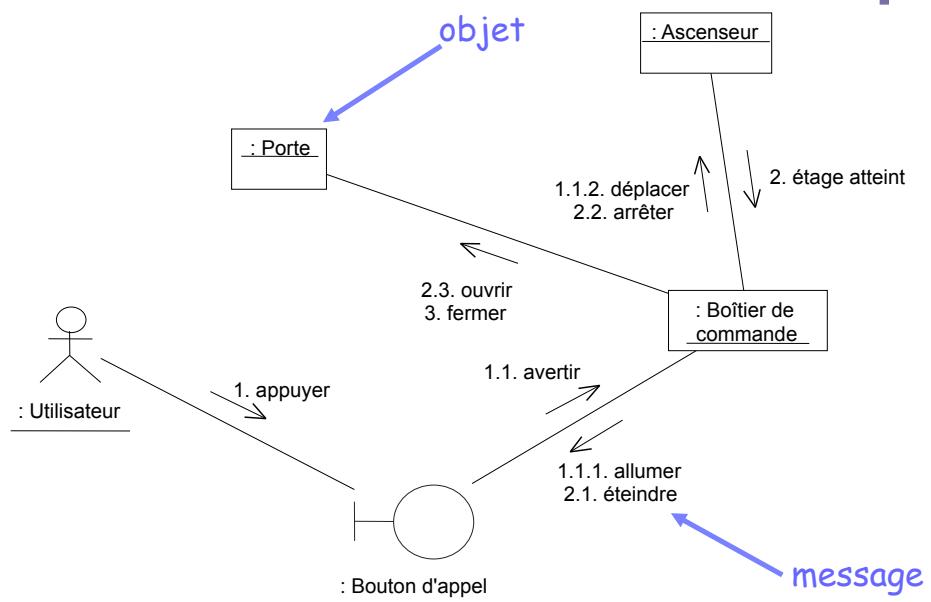
Objectif

- Représenter les interactions entre les objets de manière spatiale.

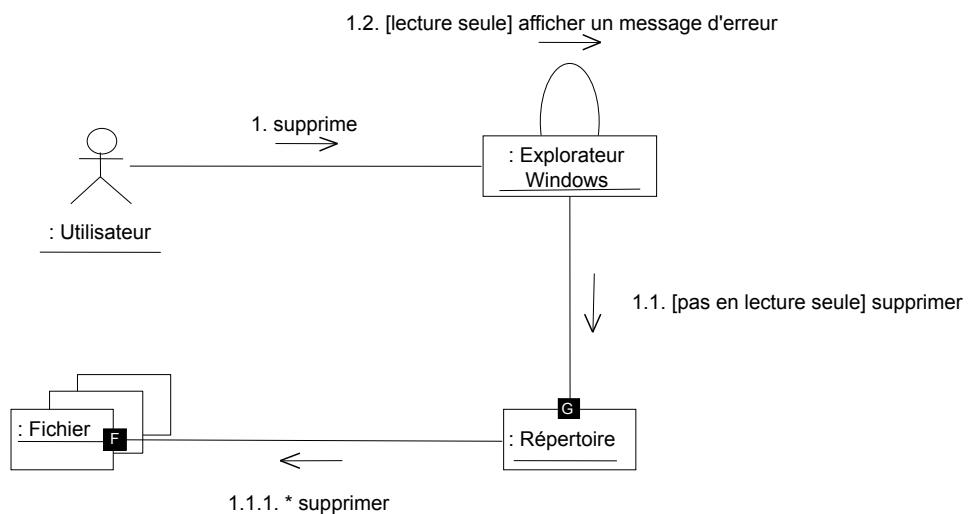
© 2013 zam@dauphine.fr



Exemple



© 2013 zam@dauphine.fr

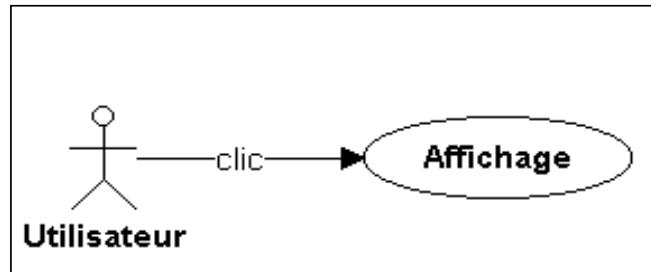


© 2013 zam@dauphine.fr



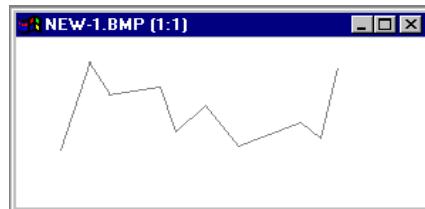
Exemple multi-diagrammes

•



```

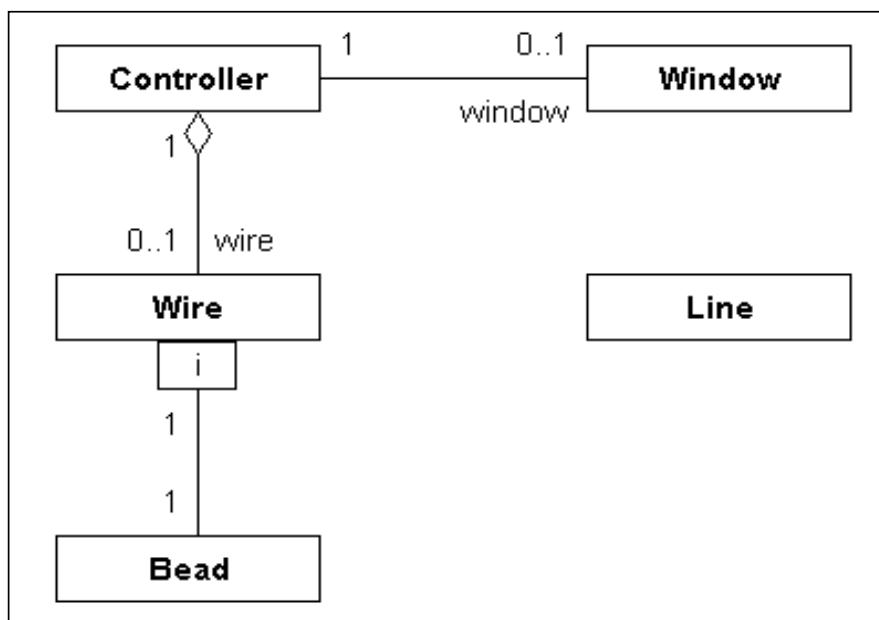
{ (3;5) , (4;3) , (5;4) ,
  (6;2) , (8;3) ,
  ...
  (22;7) }
  
```



© 2013 zam@dauphine.fr



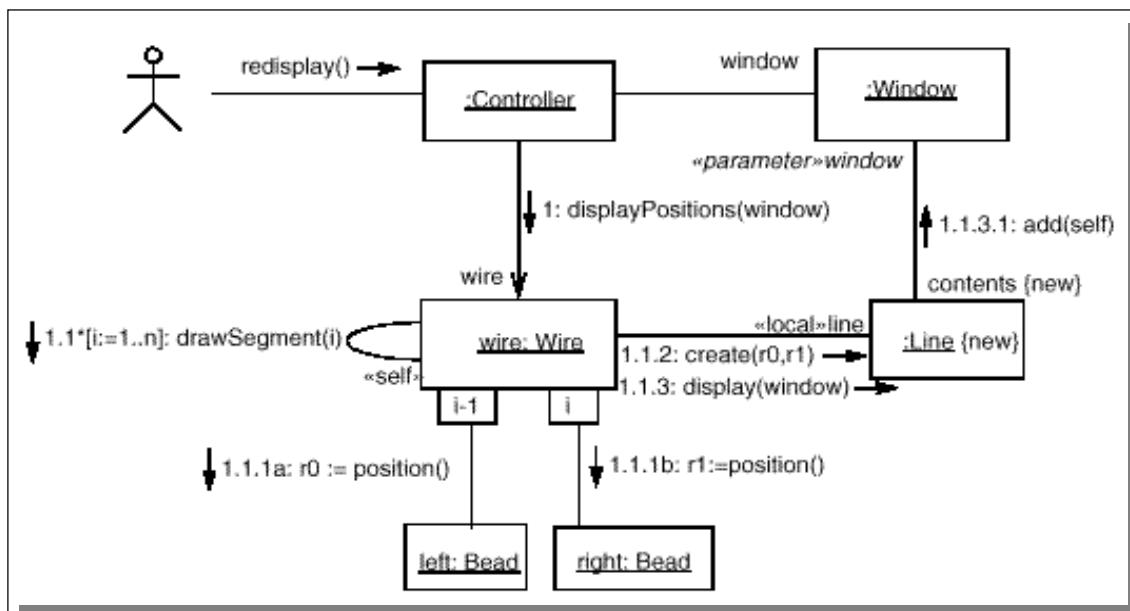
Exemple



© 2013 zam@dauphine.fr



Diagrammes de collaboration

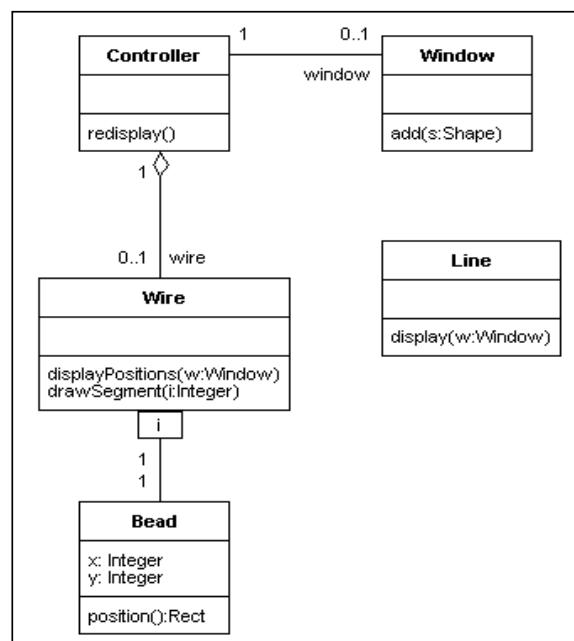


[NotUML p. 90]

© 2013 zam@dauphine.fr



Exemple



© 2013 zam@dauphine.fr

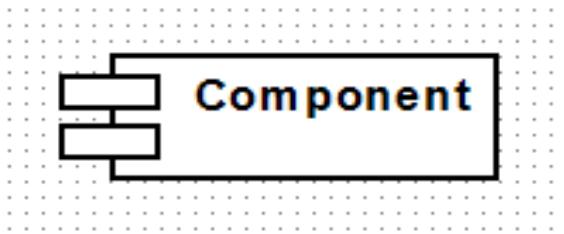


Diagramme des composantes



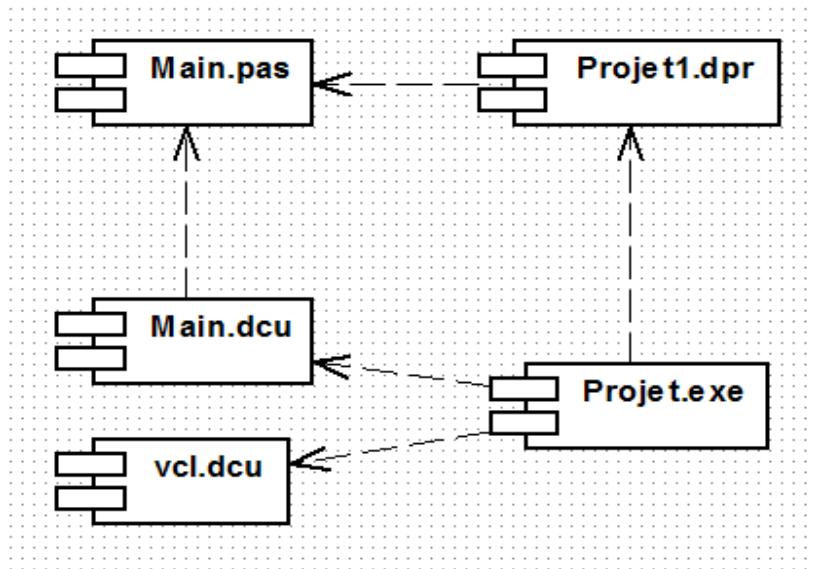
Diagramme des composantes

- Montrer les composantes de l'application et leurs dépendances.
- Composante
 - Fichier source, fichier binaire, exécutable, script, base de données, fichier de cfg, etc.





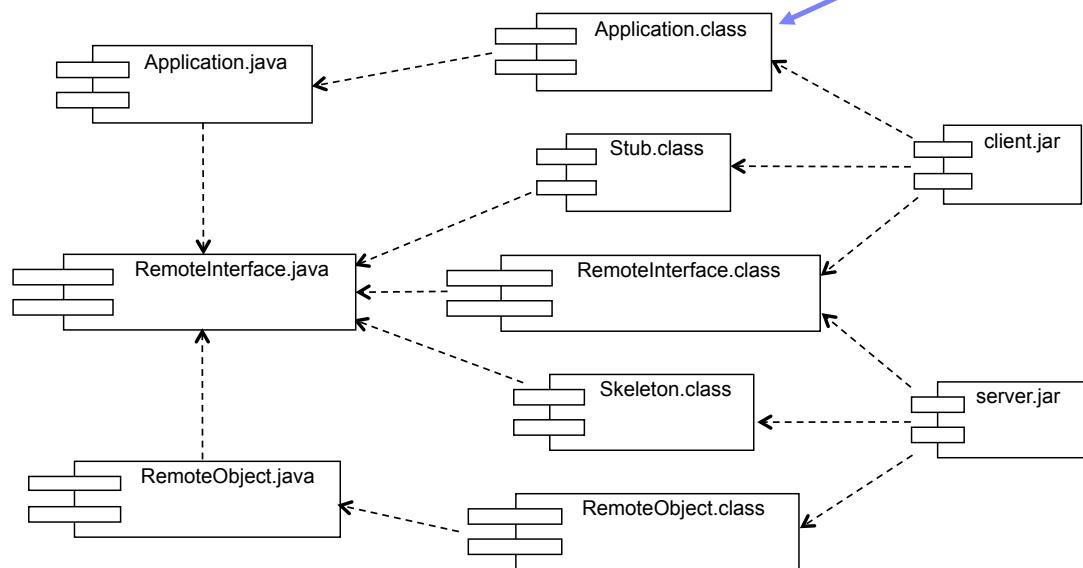
Composantes et dépendances



© 2013 zam@dauphine.fr



Exemple java



© 2013 zam@dauphine.fr



Diagramme de déploiement



Diagramme de déploiement

- Montrer les nœuds du réseau, leurs connections et préciser les composantes que ceux-ci hébergent.

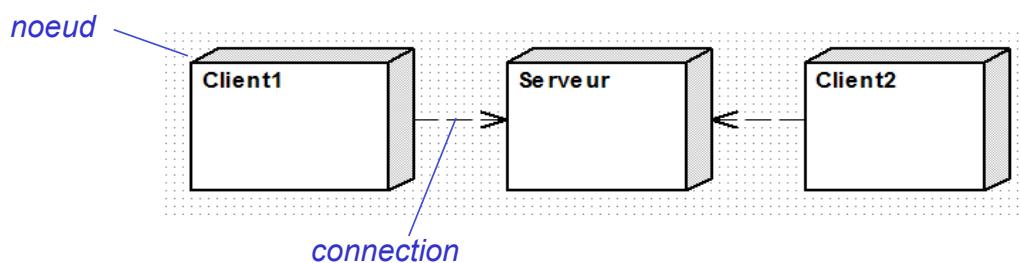
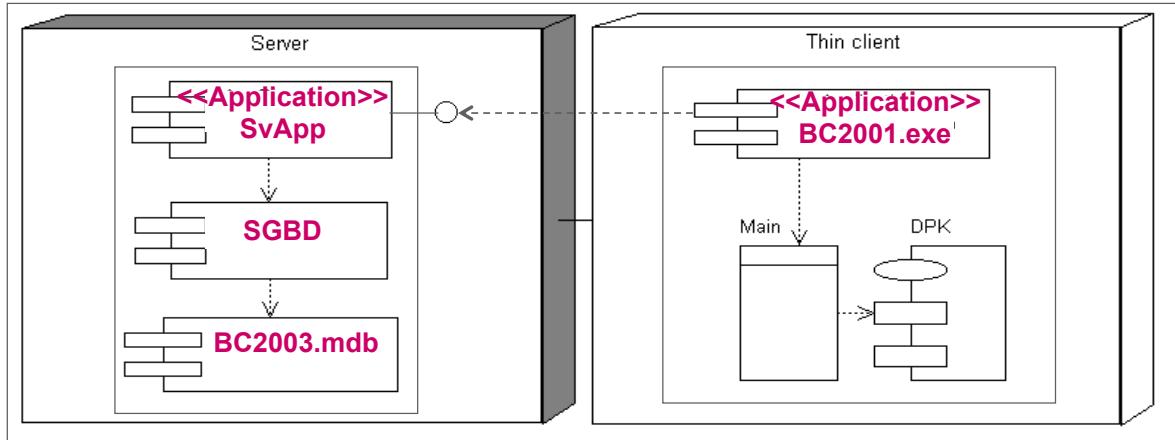




Diagramme d'implantation

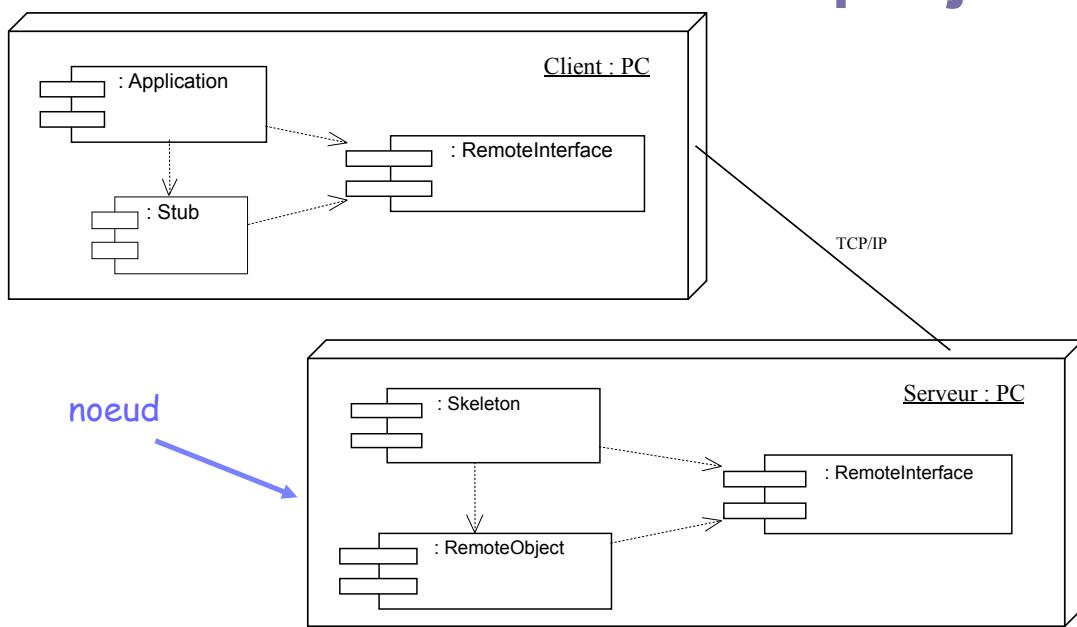
- Composantes et déploiement / noeuds



© 2013 zam@dauphine.fr



Exemple java



© 2013 zam@dauphine.fr



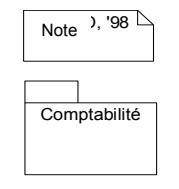
Conclusion (première partie)

Bilan et perspectives



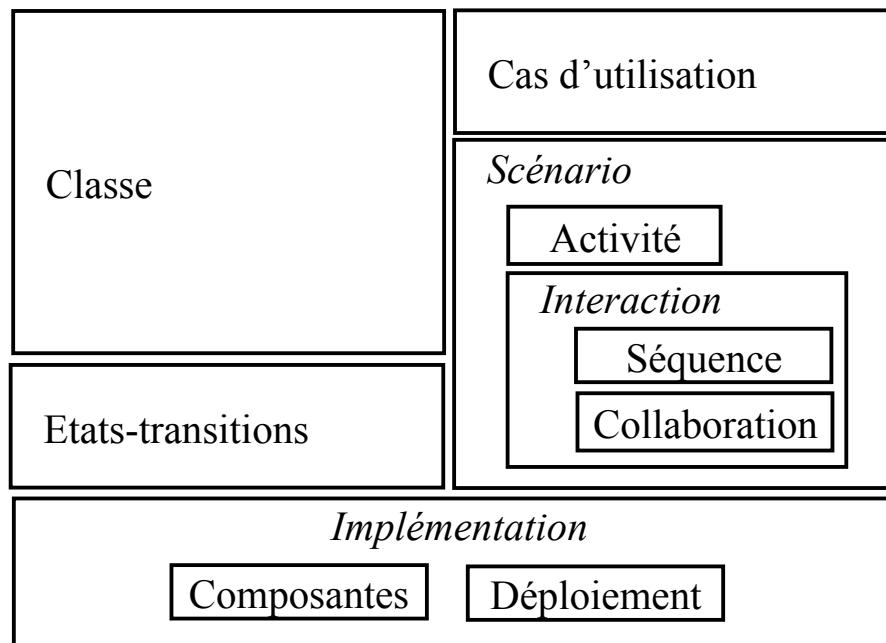
Concepts généraux

- Note
- Package
- Dépendance
- Contrainte
- Stéréotype
 - mécanisme d'extension
 - << ami >>
- Rappel de la notation : [Blaha&Rumbaugh05]

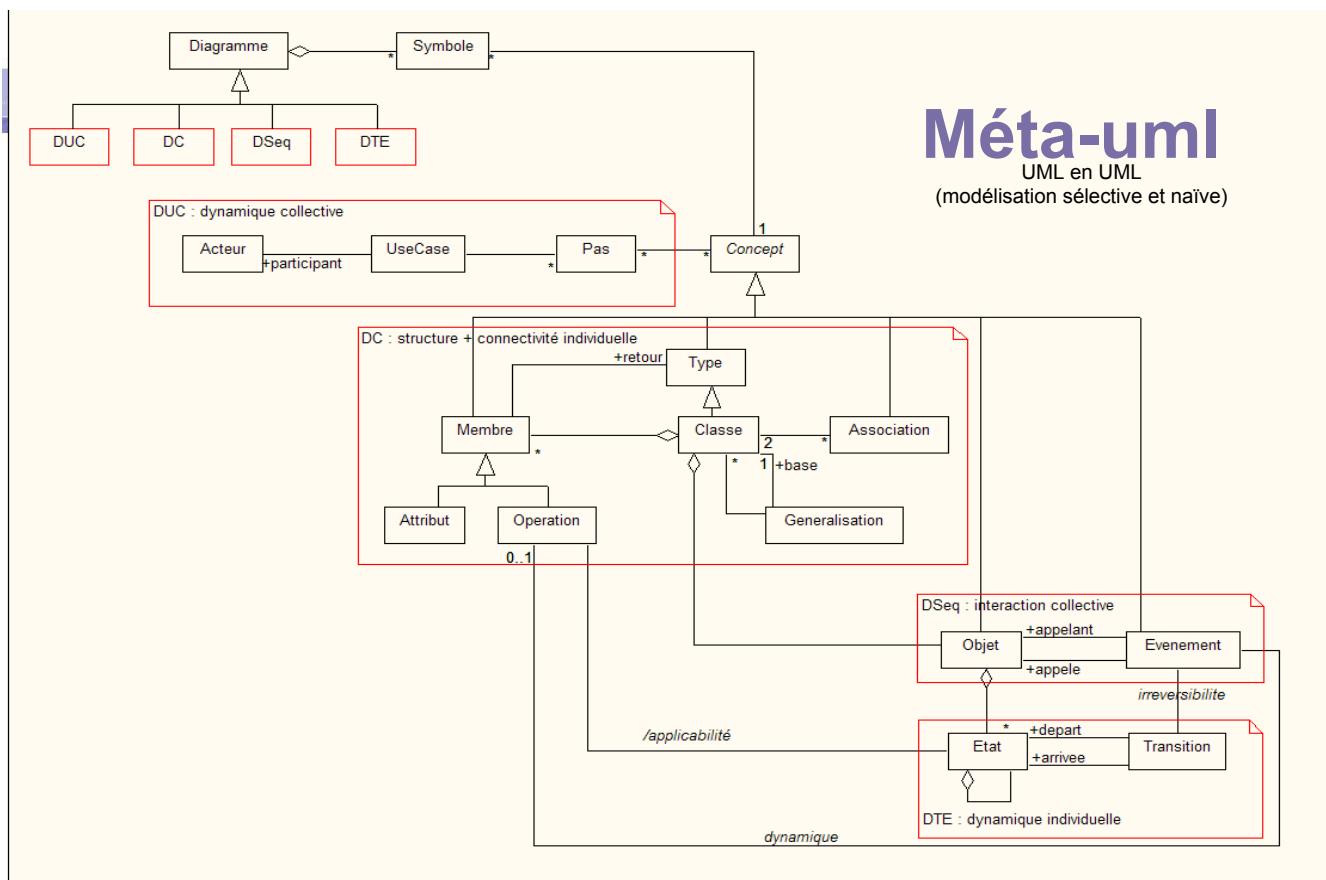




Diagrammes UML => Quel chemin ?



© 2013 zam@dauphine.fr

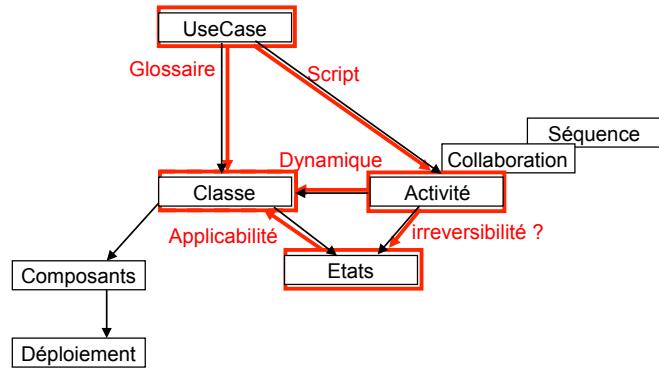


© 2013 zam@dauphine.fr

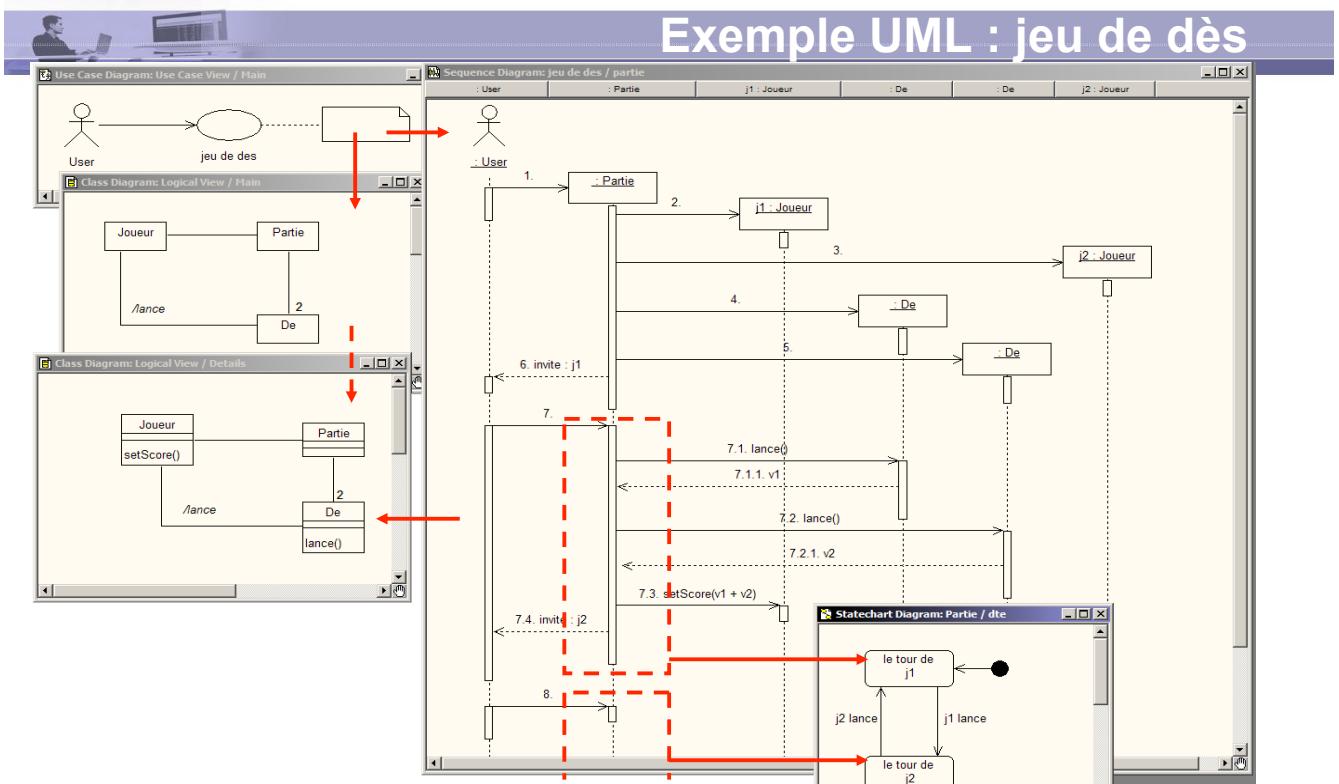


Un plan de route minimaliste

Répertorier les interactions typiques (acteurs & use cases / scripts)



© 2013 zam@dauphine.fr

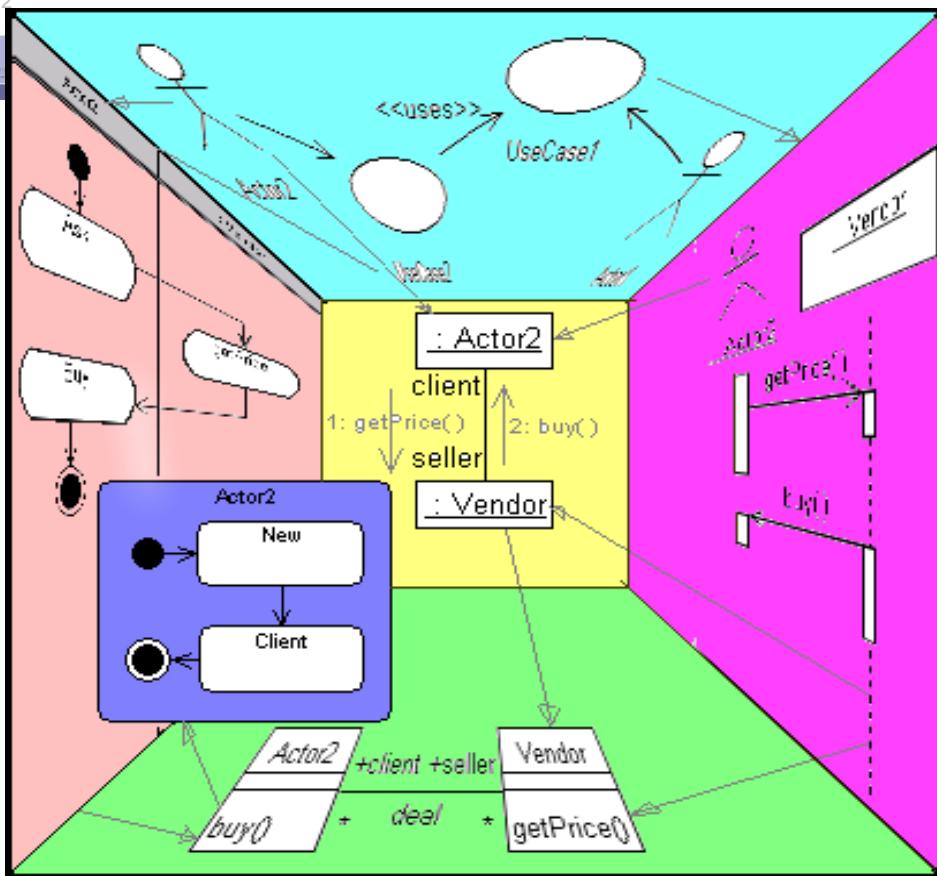


- Dynamique minimalist : chemins microscopiques

© 2013 zam@dauphine.fr



ObjectAcademy

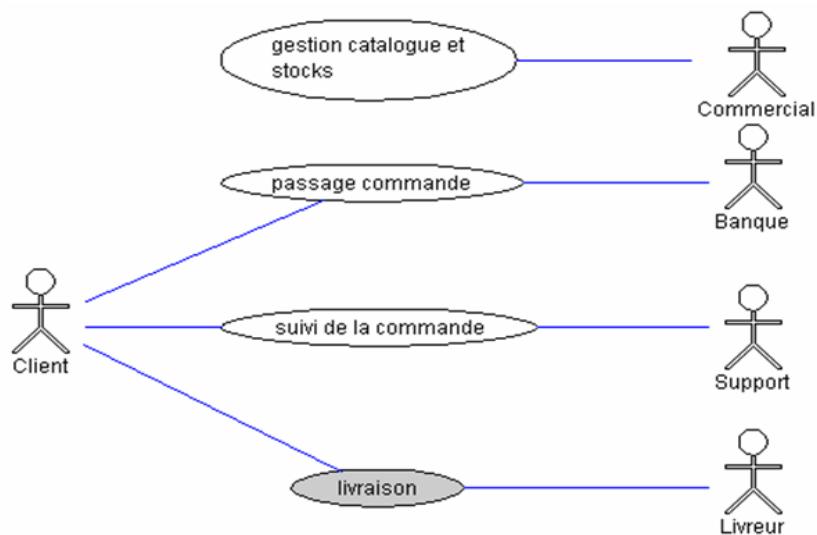


Exercice de synthèse

(et feuille de route pour modéliser léger)



WebMarchand



© 2013 zam@dauphine.fr



Feuille de route (1/2)

On souhaite la modélisation d'un système de commerce en ligne

1. Diagramme de cas d'utilisation

- donner la description textuelle des scénarii des cas d'utilisation
- attention, la livraison (en gris) est considérée en dehors du domaine cible
- définir un glossaire (par exemple, dans une note)

2. Diagramme de classe simplifié

- à partir du glossaire et des informations des scénarii,
- definir les éléments les plus importants :
- classes, attributs et associations

3. Définir un diagramme d'activité par cas d'utilisation

- représenter graphiquement les scénarii des cas d'utilisation

© 2013 zam@dauphine.fr



Feuille de route (2/2)

4. Affiner le diagramme de classes

- définir tous les éléments nécessaires :
- classes, attributs, associations, multiplicités, généralisations
- vérifier l'usage de ces éléments dans les scénarii des cas d'utilisation

5. Affiner le diagramme d'activités

- définir des couloirs d'activité en les associant aux acteurs ou aux classes
- représenter quelques "objets clé" afin de réguler le déroulement des scénarii et proposer des états adaptés

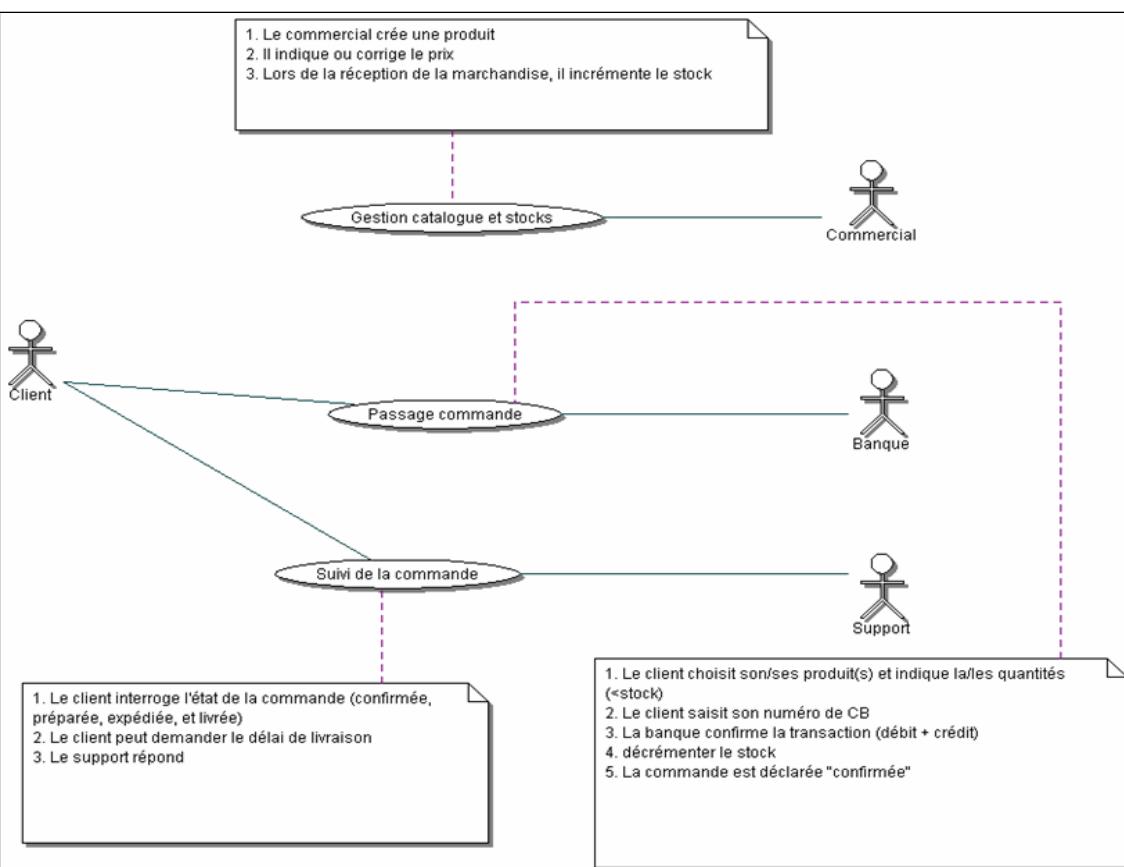
6. Affiner encore le diagramme de classes

- affectez la responsabilité des activités concernées par les classes sous la forme d'opérations

7. Diagramme de transition d'états

- à partir des états des objets proposés dans le diagramme d'activités, définir au moins un diagramme de transition d'états en précisant : états, transitions, événements, conditions, actions, activités, sousétats, etc.

© 2013 zam@dauphine.fr



© 2013 zam@dauphine.fr



Partie 2 - Repères agiles

Scrum



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value :

Individuals and interactions over *processes and tools*
Working software over *comprehensive documentation*
Customer collaboration over *contract negotiation*
Responding to change over *following a plan*

That is, while there is value in the items on the right,
we value the items on the left more.

Kent Beck, Mike Beedle, Arie van Bennekum, **Alistair Cockburn**, Ward Cunningham, **Martin Fowler**,
James Grenning, Jim Highsmith, **Andrew Hunt**, **Ron Jeffries**, Jon Kern, Brian Marick, Robert C. Martin,
Steve Mellor, **Ken Schwaber**, **Jeff Sutherland**, **Dave Thomas**

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

<http://agilemanifesto.org/>

© 2013 zam@dauphine.fr



Scrum

- I. Historique et motivation
- II. Rôles, planification & exigences
- III. Adoption progressive et ROI
- IV. Références

© 2013 zam@dauphine.fr



I. Historique

- **Origines**
 - Takeuchi et Nonaka, *The New New Product Development Game*, 1986
=> rugby [scrum = mêlée], ingénierie concurrente
 - James Martin, *RAD*, 1991
=> itératif, incrémental (vs. cascade)
- **Naissance**
 - Ken Schwaber et Jeff Sutherland, OOPSLA`95
=> Controlled Chaos Software Development, 1996
 - Scrum = un processus agile => organisation vs technique
- **Mariage & célébrité**
 - Membre de AgileAlliance.org
=> Concurrent vs complément de XP ?

© 2013 zam@dauphine.fr



Motivation

- 60 % des spécifications changent avant la fin. Comment planifier ? Une commission du changement ?
- Inspiration : «lean» / Toyota : prototype, bottom up Accion. Biologie évolutioniste. iRobot/MIT. Un system sans connaissances/données, mais avec des capteurs (le monde est une BD).
- Google, 2001 : managers licenciés ! Equipes auto-organisées de 3 à direction tournante. Corrections à tout moment, même en production. Droit à l'erreur et confiance. Tout obstacle considéré nocif doit être écarté.

© 2013 zam@dauphine.fr



Structures croissantes

- Autocratique / coercitif
un patron, râteau, ...
- Organique / responsabilisant la base
Google Ad Words : 15-20 p / 5 endroits
Yahoo : consultants / 300 pages

© 2013 zam@dauphine.fr



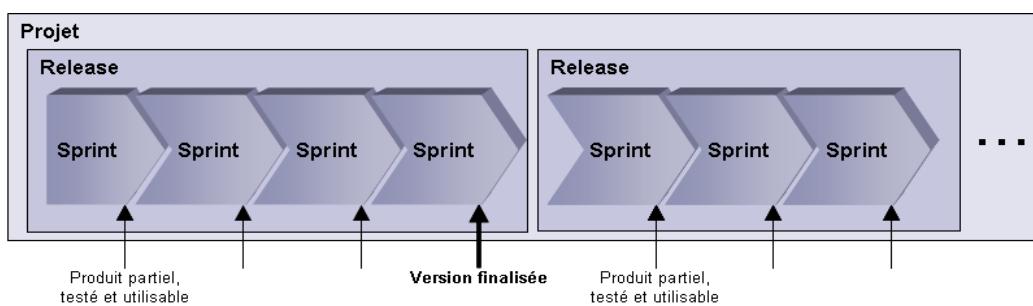
II. Rôles

- Directeur du produit (product owner) établit les priorités
- Équipe => autogérée, efficacité, qualité spontanée
- Scrum master : facilitateur, fourniture d'outils, suivi avancement
- Autres intervenants (stakeholders)

© 2013 zam@dauphine.fr



Planification



- 3 niveaux de planification :
 - projet 1 -- * release : livraisons de versions d'un produit partiel
 - release 1 -- * (4..12) sprint : itération, 3-4 semaines, testable
 - quotidien : standup Scrum Meeting

© 2013 zam@dauphine.fr



Exigences

- L'unité de spécification / exigence : l'UserStory
- Backlog (carnet des commandes à traiter)
=> ensemble d'exigences à implémenter : fonctionnelles, technologiques, pb
- Estimation : nb de «points». Point = unité arbitraire qui devient un repère. Fib : (1,2,3,5,8,13)
- Burndown chart (principale raison / célébrité) quantité de travail (points) restante, jour par jour.
- Vélocité : la pente

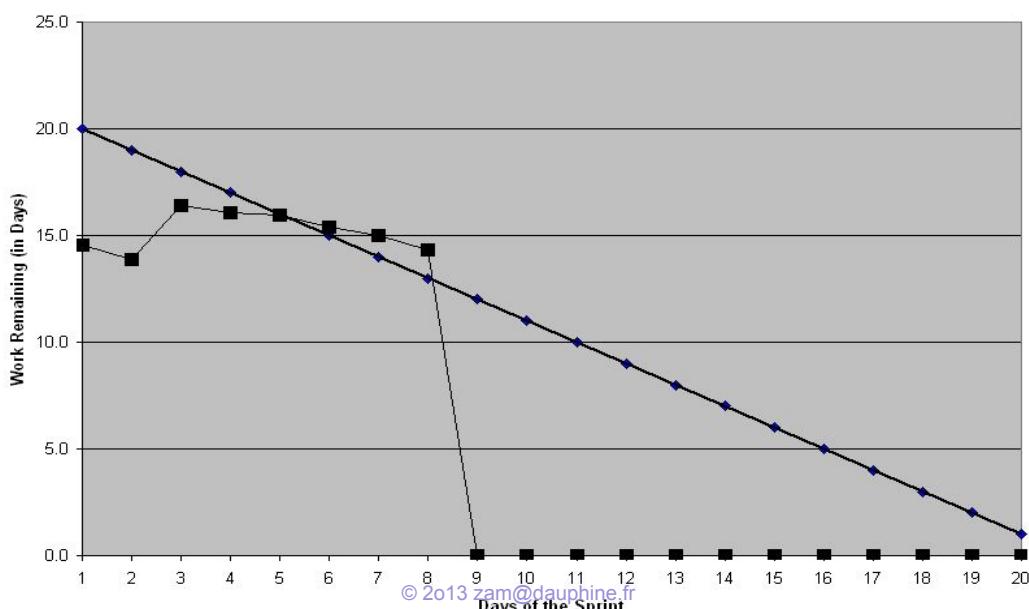
© 2013 zam@dauphine.fr



Exemple

<http://www.devx.com/enterprise/Article/34306>

Burndown Graph Shown as Days



© 2013 zam@dauphine.fr



Et les bugs ?

- Quels bugs ? ;-)
=> Defect backlog & Sprint(s) «Kill bugs»
- Scrum quotidien :
Qu'ai-je fait hier ?
Que ferai-je aujourd'hui ?
Quels problèmes ai-je rencontré ?

© 2013 zam@dauphine.fr



III. Adoption progressive

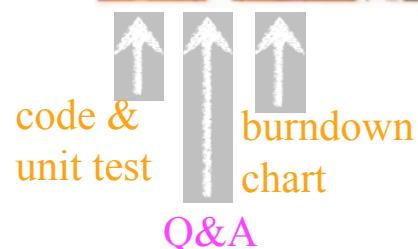
- **Retour a Google AdWords**
 - I. Product/release backlog, burndown chart, sans sprints
 - II. Q&A ? Dépendances cachées => Réunion quotidienne
 - III. Retards => 80 % travail effectué, 20% fonctionnalités terminées :-(=> minimiser le « travail en cours !»
- % spike, essais : selon la nature du projet, ex Palm > 50%
- => itérations plus fréquentes, ex Nokia <= 6 semaines

© 2013 zam@dauphine.fr



Suivi

- Tableau des tâches
- Pas d'attribution : volontariat



© 2013 zam@dauphine.fr



Tableau des tâches

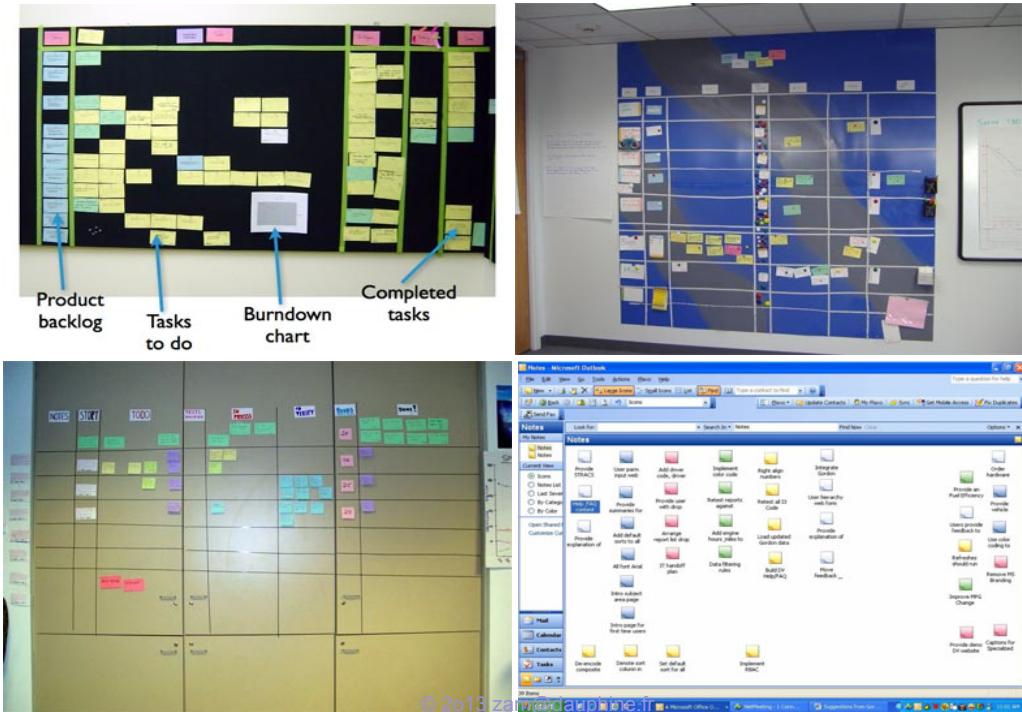
Story	To Do		In Process	To Verify	Done
As a user, I... 8 points	Code the... 9	Test the... 8	Code the... DC 4	Test the... SC 6	Code the... Test the... SC 8
	Code the... 2	Code the... 8	Test the... SC 8		Test the... SC Test the... SC
	Test the... 8	Test the... 4			Test the... SC 6
As a user, I... 5 points	Code the... 8	Test the... 8	Code the... DC 8		Test the... SC Test the... SC
	Code the... 4	Code the... 6			Test the... SC 6

www.mountaingoatsoftware.com

© 2013 zam@dauphine.fr



Suivi d'avancement (St Jude : LavaLamps)



ROI

- Offshore => 20% d'économie
- Scrum local => 2 x (Toyota : 4 x) => 50 %
- Yahoo : Scrum Offshore



Classement scrum [JS]

- Très bien - croissance de revenus <= 400%
PatientKeeper, certains scandinaves
- Bien <= 300%
d'autres scandinaves
- Assez bien <= 150% - 200%
Systematic Software Engineering : 200%
Google : 160%
- Passable <= 0-35%
Yahoo : productivité moyenne += 35%
IDC 2008 study of Agile companies : 16% amélioration

© 2013 zam@dauphine.fr



Réactivité

- Rythme des releases (livraisons) :
 - I : 4-6 mois
 - II : 2-4 semaines
 - III : chaque semaine
- Charge de travail : évolue pendant le déroulement du projet, x3 !

© 2013 zam@dauphine.fr



CMMi vs Agilité

- Les préoccupations adressées sont différentes
 - CMMi : minimise les attentes et les répétitions
 - Agilité : maximise le feedback et les priorités
- mais pas incompatibles

© 2013 zam@dauphine.fr



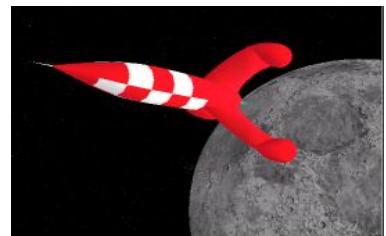
Vibliographie

- Jeff Sutherland, Scrum Tunning - Lessons learned form Scrum @ Google, 2006, 1h
http://www.youtube.com/watch?v=9y10Jvruc_Q
- Ken Schwaber, Scrum & al. @ Google, 1h
<http://www.youtube.com/watch?v=lNPeTn8fpo>
- H. Shojaee, Scrum in under 10'
<http://www.youtube.com/watch?v=Q5k7a9YEoUI>

© 2013 zam@dauphine.fr



Merci



© 2013 zam@dauphine.fr