

Point sur le rendu

- 1er rendu : vendredi 23 février 20h
 - Tous les rendus sont noté (contrôle continu)
 - Par binôme **obligatoirement**
- Document au format .doc ou .odt
 - Suivre le template **PatronGeneral**
 - Contenu :
 - Diagrammes de classes
 - Glossaire métier
 - Les diagrammes de cas d'utilisation :
 - Hiérarchie des acteurs + quiFaitQuoi
 - Documentation générée par objectteering de votre modélisation

1

Glossaire métier

- Glossaire métier = Terminologie du domaine
- Contient les termes et les définitions
- Permet de s'entendre sur les mots employés

Terme	Définition et informations
Article	Produit ou service en vente
Autorisation de paiement	Validation par un organisme externe avec garantie de paiement.
...	...

2

La Modélisation UML

Cedric Dumoulin

Compilation de présentations de :
Pierre-Alain Muller
Jean Bezivin
Bran Selic
Julie Vachon
Jeanine Leguy

3

Plan

- Pourquoi modéliser
- Bref historique
- Les éléments UML de base
- Les diagrammes statiques
- Conseils et Exemple
- Les diagramme dynamiques (dans un autre cours)

4

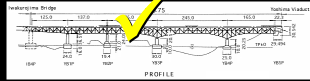
What Engineers Do

IBM

- ◆ Before they build the real thing...



...they first build models ...and then learn from them



5

IBM Software Group Rational

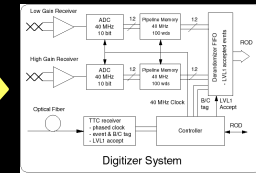
Engineering Models

IBM

- ◆ Engineering model:
A reduced representation of some system



Modeled system



Model

- ◆ Purpose:
To help us understand a complex problem or solution
To communicate ideas about a problem or solution
To drive implementation

6

IBM Software Group Rational

Characteristics of Useful Models

IBM

- ◆ Abstract
 - Emphasize important aspects while removing irrelevant ones
- ◆ Understandable
 - Expressed in a form that is readily understood by observers
- ◆ Accurate
 - Faithfully represents the modeled system
- ◆ Predictive
 - Can be used to derive correct conclusions about the modeled system
- ◆ Inexpensive
 - Much cheaper to construct and study than the modeled system

To be useful, engineering models must have all of these characteristics!

7

IBM Software Group Rational

How Models are Used

IBM

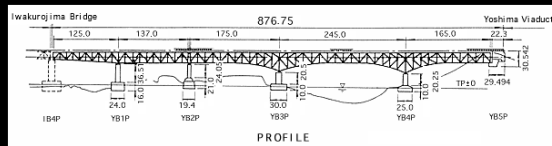
- ◆ To detect errors and omissions in designs before committing full resources to full implementation
 - Through (formal) analysis and experimentation
 - Investigate and compare alternative solutions
 - Minimize engineering risk
- ◆ To communicate with stakeholders
 - Clients, users, implementers, testers, documenters, etc.
- ◆ To drive implementation

8

IBM Software Group Rational

A Problem with Models

IBM



Semantic Gap due to:

- Idiosyncrasies of actual construction materials
- Construction methods
- Scaling effects
- Skill sets
- Misunderstandings

Can lead to serious errors and discrepancies in the realization

9

IBM Software Group | Rational

COA

Définitions

Une modélisation est une activité qui permet d'appréhender un système complexe en vue d'en déduire ses caractéristiques en tenant compte de différents **points de vue**.

Une modélisation conduit à un ensemble de **modèles**.

Un modèle est une simplification de la réalité

Un modèle est une abstraction sémantiquement close

Un modèle peut prendre différentes formes: schéma, texte, ...

Un modèle peut être statique (organisation des éléments) ou dynamique (comportement des éléments)

- 10 -

COA

Pourquoi modéliser ?

▸ **Pour comprendre.**

On modélise des systèmes complexes car on ne peut pas comprendre de tels systèmes dans leur intégralité :

La modélisation permet de reculer les limites humaines en se focalisant sur un aspect à la fois.

▸ **Pour borner le champ d'investigation**

On n'a pas à comprendre tout sur tout.

▸ **Pour communiquer**

Nécessité d'un langage commun, précis sans ambiguïté, en minimisant au maximum les interprétations possibles. Permet de tenir compte du point de vue et de la connaissance de chacun.

- 11 -

COA

Les 4 grands principes de la modélisation

▸ **Le choix des modèles**

Le choix des modèles à développer influence la façon dont un problème est abordé et dont une solution est trouvée.

▸ **Niveau de précision (granularité)**

Chaque modèle peut être exprimé à différents niveaux de précision

▸ **Modèle et réalité**

Les meilleurs modèles sont ceux rattachés à la réalité

▸ **Point de vue**

Aucun modèle n'est suffisant. Tout système non trivial doit être abordé à travers différents points de vue relativement indépendants

- 12 -

UML COA

Historique

- 13 -

Historique : Évolution de UML COA

UML d'après les «amigos»

0.8 -> 0.9
0.9 -> 0.91 -> 1.0
1.0 -> 1.1 -> 1.2 -> 1.3 -> 1.4

→ 1.5
→ 2.0
→ x.y

Comprendre la logique d'évolution d'UML.

- 14 -

De Rational à l'OMG COA

1.1 -> 1.2 -> 1.3 -> 1.4 -> 1.5 -> 2.0

?

Soumission de UML 1.0 à OMG pour adoption (janvier 1997).

UML 1.0

UML partners expertise

UML 0.9 & 0.91 (juin 96 - oct. 96)

OOPSLA'95 Unified Method 0.8

Booch 93, OMT-2

Booch 91, OMT-1

Autres méthodes (≈ 50)

OOSE

Industrialisation

Standardisation

Unification

Fragmentation

?

- 15 -

Devant et derrière, Avant et après ... COA

Méthode = Langage(s) + Démarche + Outils

procédés industriels de production de logiciels et de systèmes.

OMT SA/RT SADT
ERD Merise
DFPD
JSD etc.

- 16 -

COA

Et un langage unique, un!



La définition du formalisme UML ne résulte pas d'un processus innovant de recherche mais d'un processus consensuel de stabilisation des pratiques Industrielles éprouvées.

UML n'est que le reflet fidèle des pratiques majoritaires utilisées vers la fin des années 2000 par la profession.

UML ne vise pas l'innovation, mais la consensualité.

Il y a deux façons de réaliser un consensus: à minima (par intersection) ou à maxima (par union).

Ces deux tendances se retrouvent dans les groupes d'influence qui gèrent l'évolution du langage (vendeurs d'AGL, etc.)

La tendance maximaliste a souvent été majoritaire (!)



- 17 -

COA

UML n'est pas un projet de recherche

"In short: the time for experimentation is past; the time for stability and use is now."

Grady Booch
Chief Scientist
Rational Software Corporation

- 18 -

COA

Evolution de la terminologie

Booch	Classe	Utilise	Hérite	Contient
Coad	Classe/Objet	Connexion d'instance	Gen/Spec	Tout/Partie
Jacobson	Objet	Accointance Association	Hérite	ConsisteEn
Odell	Objet/Type	Relation	Sous- type	Composition
Rumbaugh	Classe	Association	Généralisation	Aggrégation
Shlaer/Mellor	Objet	Relation	Sous- type	N/D
UML	Classe	Association	Généralisation	Aggrégation

- 19 -

COA

Quelques notations de cardinalité

Booch. 1				
Coad				
Rumbaugh				
UML				

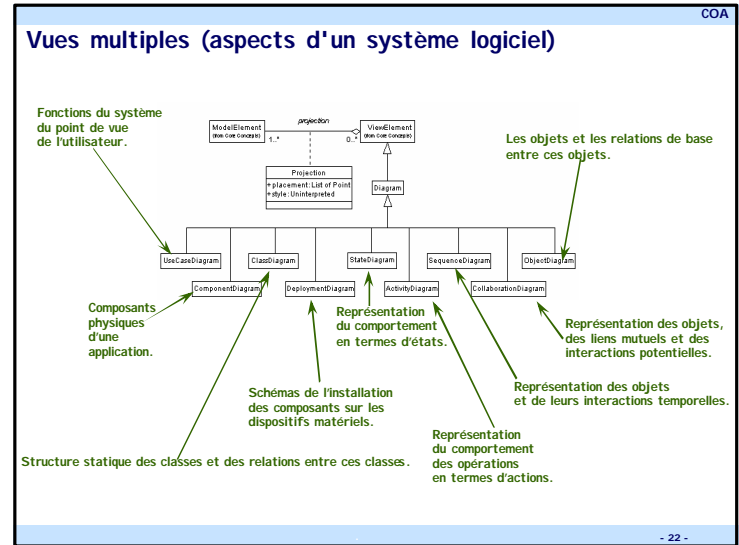
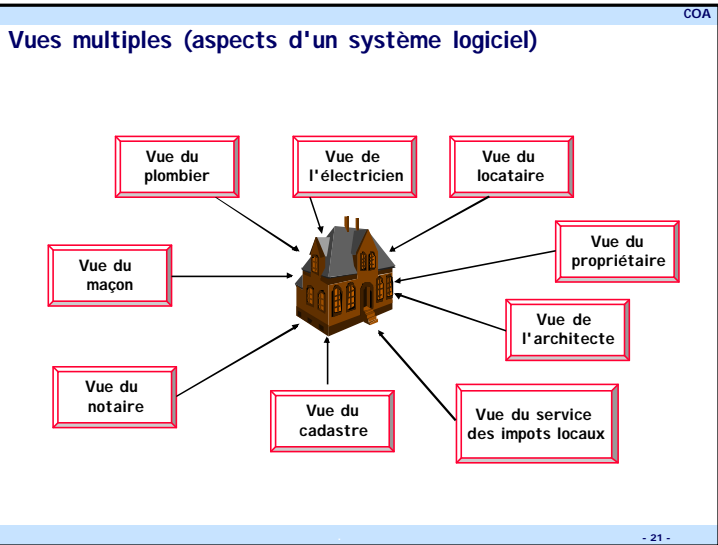
un A toujours associé avec un B.

un A toujours associé avec un ou plusieurs B.

un A associé avec zéro ou un B.

un A associé avec zéro, un ou plusieurs B.

- 20 -



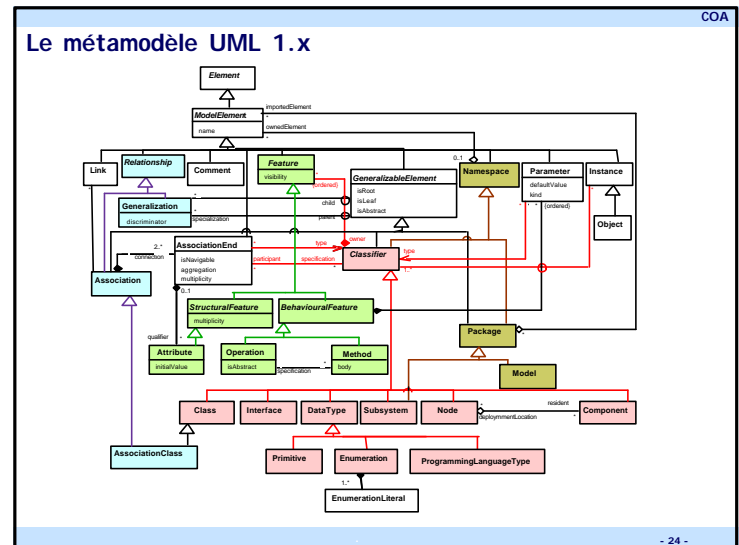
COA

Qualités d'UML

- La définition d'UML s'appuie sur un méta modèle décrit en UML
- validation et précision de la notation
- Définition relativement rigoureuse qui permet un contrôle. (Un peu à la manière des langages de programmation, le contrôle n'empêche pas d'écrire n'importe quoi !)
- Permet la mise en place du paradigme Objet
- langage graphique relativement compréhensible par tous
- manipulable par des outils (modeleurs UML)
- utilisable dans tous les domaines (télécom, santé, banques, transport, ...) et pour tout type de système (du SI aux Intranets en passant par les systèmes embarqués temps réel. mais également des systèmes non informatiques comme par exemple l'organisation du système qualité d'un hôpital).

UML est du domaine public, tout le monde peut l'utiliser.

- 23 -



Défauts d'UML

- ⌚ UML est un langage, pas une méthode.
- En particulier, on ne nous dit pas comment utiliser UML, l'utilisation des différents outils à travers le cycle de vie d'un développement n'est pas définie. Une méthode s'impose!
- ⌚ UML manque de précision => OCL
- ⌚ UML est difficilement manipulable sans outil.

L'utilisation d'UML nécessite une très grande rigueur qui peut être occultée par le côté visuel de la notation

Modèle décrit en UML

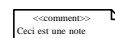
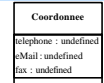
- Il est composé de **briques de base**
- Il doit respecter les **règles UML**
- Il utilise les **mécanismes communs** d'UML

Les composants UML

- Les briques de base
 - Les éléments
 - Les relations
 - Les diagrammes
- Les règles UML
 - Les règles sont définies dans un manuel de référence. Tout outil permettant de manipuler UML doit se conformer à ces règles.
 - Un modèle est dit **bien formé** s'il est conforme à ces règles
- Les mécanismes communs
 - Stéréotypes: méta classification permet de visualiser des ressemblances entre briques de bases d'un même « genre »
 - Les valeurs 'étiquetées'

Les Eléments

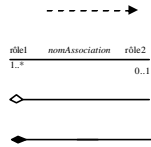
- Les éléments structurants
 - Les classes
 - Les cas d'utilisation
 - Les composants
- Les élément comportementaux
 - Les interactions
 - Les automates
- Les éléments de regroupement
 - Les paquetages
- Les éléments de documentation
 - Les notes
 - Les contraintes



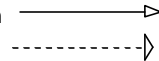
Les Eléments

• Les relations

- de dépendance
- d'association



- de généralisation
- de réalisation



Visite guidée rapide du langage

- Il y a 9 diagrammes en UML 1.x.
 - Les diagrammes de classes
 - Les diagrammes d'objets
 - Les diagrammes de cas d'utilisation
 - Les diagrammes de séquence
 - Les diagrammes de collaboration
 - Les diagrammes d'états/transitions
 - Les diagrammes d'activités
 - Les diagrammes de composants
 - Les diagrammes de déploiement



avec la contribution
de Pierre-Alain Muller

Diagrammes en UML 2.x.

• Structural Modeling Diagrams (6)

- **Package diagram** - are used to divide the model into logical containers, or 'packages', and describe the interactions them a high level.
- **Class or Structural diagrams** define the basic building blocks of a model: the types, classes and general materials used to construct a full model.
- **Object diagrams** show how instances of structural elements are related and used at run-time.
- **Composite Structure diagrams** provide a means of layering an element's structure and focusing on inner detail, construction and relationships.
- **Component diagrams** are used to model higher level or more complex structures, usually built up from one or more classes, and providing a well defined interface.
- **Deployment diagrams** show the physical disposition of significant artifacts within a real-world setting.

• Behavioral Modeling Diagrams (7)

- **Use Case diagrams** are used to model user/system interactions. They define behavior, requirements and constraints in the form of scripts or scenarios.
- **Activity diagrams** have a wide number of uses, from defining basic program flow, to capturing the decision points and actions within any generalized process.
- **State Machine diagrams** are essential to understanding the instant to instant condition, or "run state" of a model when it executes.
- **Sequence diagrams** show the network, and sequence, of messages or communications between objects at run-time, during a collaboration instance.
- **Swimlane diagrams** are closely related to communication diagrams and show the sequence of messages passed between objects using a vertical timeline.
- **Timing diagrams** fuse sequence and state diagrams to provide a view of an object's state over time, and messages which modify that state.
- **Interaction Overview diagrams** fuse activity and sequence diagrams to allow interaction fragments to be easily combined with decision points and flows.

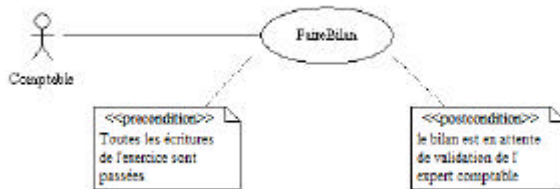
Diagrammes de cas d'utilisation

Use Cases

Le diagramme de cas d'utilisation

Il est composé de

- Cas d'utilisation
- Acteurs
- de relations de dépendance, de généralisation, d'association
- paquetage
- notes
- contraintes



Les diagrammes de cas d'utilisation

Servent à identifier ce que fait un système et les acteurs qui gravitent autour du système
Outil de communication entre le MOA et la MOE (système logiciel).

On l'utilise pour

• Modéliser le contexte d'un système

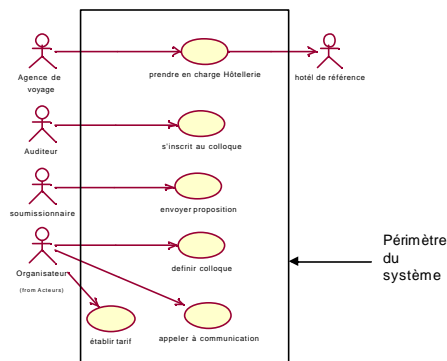
C'est à dire qui (ou quoi) est concerné par le système. Ceci se traduit concrètement par la détermination des **acteurs**, humains ou non, leurs rôles par rapport au système

• Pour modéliser ce que fait un système

Pour répertorier tous les **services** qu'on peut attendre d'un système

Et a contrario tout ce qu'il n'est pas sensé faire, c'est à dire définir le **périmètre** du système.

Illustration



Les diagrammes de cas d'utilisation

Un diagramme de cas d'utilisation ne représente jamais le déroulement d'un système, il dit ce qu'il fait (ou fera), jamais comment il le fait (fera).

Un cas d'utilisation est un **service** (fonctionnalité) du système, il représente une unité cohérente identifiable de l'extérieur du système. Il permet de définir un comportement sans révéler quoi que ce soit sur la réalisation.

Les cas d'utilisation peuvent être décorés avec des assertions (pré et post conditions).

On distingue les cas d'utilisation **métier** (existent indépendamment du système logiciel).

et les cas d'utilisation **applicatifs**

Classes

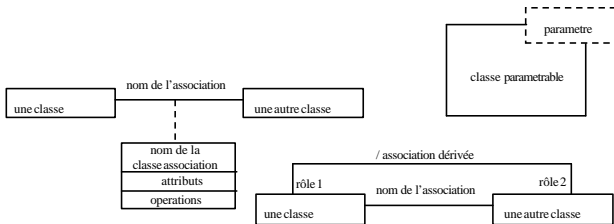
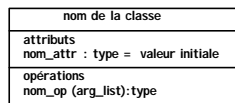
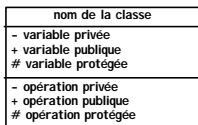
Diagrammes de classes

Ils servent à modéliser l'aspect statique d'un système

Il est composé de :

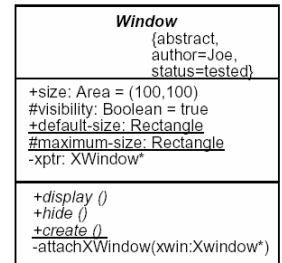
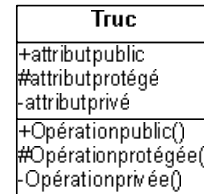
- Classes
- Paquetages
- Relations
- Notes ...

Diagrammes de classe.



Visibilité des propriétés

- **Public +**
Visible à l'extérieur de la classe
- **Protégé #**
Visible seulement par les descendants
- **Private -**
Visible à l'intérieur des méthodes
- **Souligné**
Variable/opération de classe



COA

Variantes de classes

- Classe emboîtée
 - Déclarée dans la portée lexicale d'une autre classe.

```

classDiagram
    class DeclaringClass
    class NestedClass
    DeclaringClass -- NestedClass
  
```

- 41 -

COA

Variantes de classes

- Classe paramétrée
 - Modèle de classe
 - Paramètres Formels génériques (types, opérations...)
- Classe utilitaire
 - Espace de nommage, groupement de variables + opérations

```

classDiagram
    class FArray {
      T,k: Integer
      k..k
      T
    }
    class AddressList
    FArray ..> AddressList : «bind» (Address,24)
    class MathPak {
      «utility»
      sin (Angle): Real
      cos (Angle): Real
      sqrt (Real): Real
      random(): Real
    }
  
```

- 42 -

COA

Variantes de classes

- Interfaces
- Spécification des opérations visibles
 - Classes, paquetage, composants
- Contrat sans implémentation
 - Pas d'attributs, de méthodes, de relations
 - Peut être cible d'une association unidirectionnelle
- Deux représentations graphique
 - Stéréotype, lollipop

```

classDiagram
    class POSterminal
    class Store {
      -storeId: Integer
      -POSlist: List
      +create()
      +login(Username, Password)
      +find(StoreId)
      +getPOStotal(POSid)
      +updateStoreTotal(Id, Sales)
      +get(Item)
    }
    class StoreInterface {
      <<interface>>
      +getPOStotal(POSid)
      +updateStoreTotal(Id, Sales)
      +get(Item)
    }
    POSterminal -- Store
    POSterminal -- StoreInterface
    Store ..> StoreInterface : «use»
  
```

- 43 -

COA

Variantes de classes

- Metaclass
 - La classe d'une classe.
- Enumération
 - Ensemble de littéraux d'énumération, ordonné.
- Powertype
 - Les instances du powertype sont des classes du modèle, s'utilise pour la méta-modélisation.

- 44 -

COA

Les relations entre classes

- L'association
- L'agrégation
- La composition
- La généralisation
- La dépendance
- L'association exprime une connexion sémantique bidirectionnelle entre classes
- Une association est une abstraction des liens qui existent entre les objets instances des classes associées

- 45 -

COA

Exemple

```

classDiagram
    class Personne
    class Homme
    class Femme
    Personne <|-- Homme
    Personne <|-- Femme
    Personne "0..1" -- "0..1" : pacséA
    Homme "0..1" -- "0..1" : mariéA
  
```

- 46 -

COA

Nommage des associations

- Indication du sens de lecture

```

classDiagram
    class Université
    class Etudiant
    Université --> Etudiant : Héberge
    Université <-- Etudiant : Etudie dans
  
```

- 47 -

COA

Nommage des rôles

- Le rôle décrit une extrémité d'une association

```

classDiagram
    class Université
    class Personne
    Université -- Personne : +Etudiant
    Université -- Personne : +Employeur
    Université -- Personne : +Enseignant
  
```

- 48 -

COA

Multiplicité des rôles

The first diagram shows a role 'travaillePour' with a multiplicity of 1 at 'Personne' and 1 at 'Compagnie'.
 The second diagram shows a role 'employé' with a multiplicity of 1 at 'Personne' and 'employeur' with a multiplicity of 1 at 'Compagnie'.
 The third diagram shows a role with a multiplicity of 1..* at 'Personne' and * at 'Compagnie'.

1 Un et un seul
 0..1 Zéro ou un
 M .. N De M à N (entiers naturels)
 * Plusieurs
 0 .. * De zéro à plusieurs
 1 .. * D'un à plusieurs

- 49 -

COA

Visibilité des rôles

- Public
- Protégé
- Private

Class A is connected to class B with a role '+Rolepublic' (public) and a multiplicity of * at B. Class A is also connected to class C with a role '#Role protégé' (protected) and a multiplicity of * at C.

- 50 -

COA

Exemple : graphe non orienté

The diagram shows three classes: 'Graphe', 'Sommet', and 'Arête'. 'Graphe' has a composition relationship with 'Sommet' (multiplicity 1 at Graphe, 0..* at Sommet) and with 'Arête' (multiplicity 1 at Graphe, 0..* at Arête). 'Sommet' has a composition relationship with 'Arête' (multiplicity 2 at Sommet, 1..* at Arête) labeled 'relie'.

- 51 -

COA

Exemple : graphe orienté

The diagram shows three classes: 'Graphe', 'Sommet', and 'Arc'. 'Graphe' has a composition relationship with 'Sommet' (multiplicity 1 at Graphe, 0..* at Sommet) and with 'Arc' (multiplicity 1 at Graphe, 0..* at Arc). 'Sommet' has a composition relationship with 'Arc' (multiplicity 1 at Sommet, 0..* at Arc) labeled 'source'. 'Arc' has a composition relationship with 'Sommet' (multiplicity 0..* at Arc, 0..* at Sommet) labeled 'destination'. A note box states: 'Tout sommet doit être relié au moins à un autre sommet.'

- 52 -

COA

Navigabilité

- étant donnée une association non décorée entre deux classes, on peut naviguer d'un type d'objet vers un autre type d'objet.
- par défaut une association est navigable dans les deux sens.
- une indication de navigabilité suggère en général qu'à partir d'un objet à une extrémité on peut directement et facilement atteindre l'un des objets à l'autre extrémité.
- lors d'une mise en œuvre programmée, ceci peut suggérer par exemple qu'un objet source mémorise une référence directe aux objets cible.

Utilisateur

+propriétaire
 1 —————> *
 MotDePasse

- étant donné un utilisateur, on désire pouvoir accéder à ses mots de passe
- étant donné un mot de passe, on ne souhaite pas pouvoir accéder à l'utilisateur correspondant

- 53 -

COA

Exemple

```

classDiagram
    Ecole "1" --> "1..*" Département : possède
    Etudiant "1..*" --|> "1..*" Département : membre
    Etudiant "*" --> "1..*" Cours : suit
    Enseignant "1..*" --> "1..*" Cours : donne
    Enseignant "0..1" --> "1..*" Département : dirigéPar
    Enseignant "1..*" --> "1..*" Département : estAffectéà
    Enseignant "0..1" --> "1..*" Cours : +directeur
  
```

- 54 -

COA

Les classes-associations

- Ajout d'attributs ou d'opérations dans la relation

```

classDiagram
    A "*" -- "*" B
    C -- "*" D
    class C {
      -a1
      -a2
      +op1()
      +op2()
    }
  
```

- 55 -

COA

Associations ternaires (et plus)

- Pas d'agrégation, pas de qualifier
- Multiplicité plus difficile à lire

```

classDiagram
    Team "*" -- "*" Year : season
    Team "*" -- "*" Player : goalkeeper
    Record -.-> {Team, Year, Player}
    class Record {
      goals for
      goals against
      wins
      losses
      ties
    }
  
```

- 56 -

COA

L'agrégation

- (losange blanc)
- Forme d'association qui exprime un couplage plus fort entre classes
- Représentation des relations
 - maître et esclaves
 - tout et parties
 - composé et composant.

- 57 -

COA

La composition

- (losange noir)
- Modélisation de la composition physique
 - Multiplicité au max de 1 du côté de l'agrégat
 - Propagation automatique de la destruction

- 58 -

COA

Hierarchies de classes

- Gérer la complexité
 - Arborences de classes d'abstraction croissante

- 59 -

COA

Les notes

- Commentaire attaché à un ou plusieurs éléments de modélisation
 - Appartient à la vue, pas au modèle
 - Peut être stéréotypée en contrainte

- 60 -

COA

Les contraintes

- Relation sémantique quelconque entre éléments de modélisation
- Exprimée en OCL (Object Constraint Language) ou en langage naturel
 - {contrainte}, inv, pre-, post-condition

A

«postcondition»
{Count > 10}

- 61 -

COA

Exemples de contraintes

Copie

Livre

{xor}

Journal

Transaction

quantité :Euro (quantité est un multiple de €5)

COA

Diagrammes d'objets

Objets

- 63 -

COA

Représentation graphique des objets

- Le nom d'un objet est souligné
 - Nom : Classe
 - Nom
 - :Classe

triangle: Polygon

center = (0,0)
 vertices = ((0,0),(4,0),(4,3))
 borderColor = black
 fillColor = white

triangle

:Polygon

triangle: Polygon

scheduler

awindow : Window

horizontalBar:ScrollBar

verticalBar:ScrollBar

surface:Pane

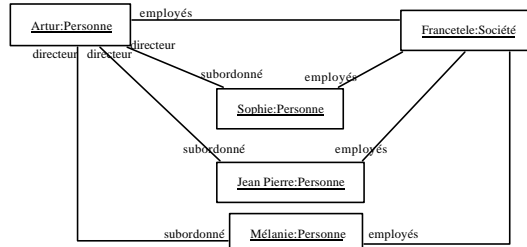
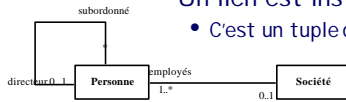
title:TitleBar

moves moves

- 64 -

Représentation des liens

- Un lien est instance d'une association
 - C'est un tuple de références vers des objets



3 types de diagrammes avec des objets

- Diagrammes d'objets (point de vue statique)
- Diagrammes d'interaction (point de vue dynamique)
 - Diagramme de séquence
 - Diagramme de collaboration
- Deux niveaux de représentation des collaborations
 - Niveau Spécification (des rôles et des messages)
 - Niveau Instance (des instances et des stimuli)

Diagrammes d'objets

- Les *diagrammes d'objets* représentent un ensemble d'objets et leurs liens. Ce sont des vues statiques des instances des éléments qui apparaissent dans les diagrammes de classes. Ils présentent la vue de conception d'un système, exactement comme les diagrammes de classes, mais à partir de cas réels ou de prototypes.
- Un diagramme d'objet est une instance d'une diagramme de classes. Les diagrammes de classes peuvent aussi contenir des objets (*objets de classes*, au sens *variables de classes*)

Paquetages

Les Paquetages

Elément de structuration par excellence, un paquetage peut contenir des paquetages, des classes, des diagrammes,

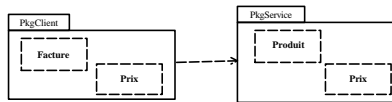
Un paquetage est un **conteneur** qui définit un **espace de nommage** et qui permet de voir (de cacher) son contenu.

Convention :

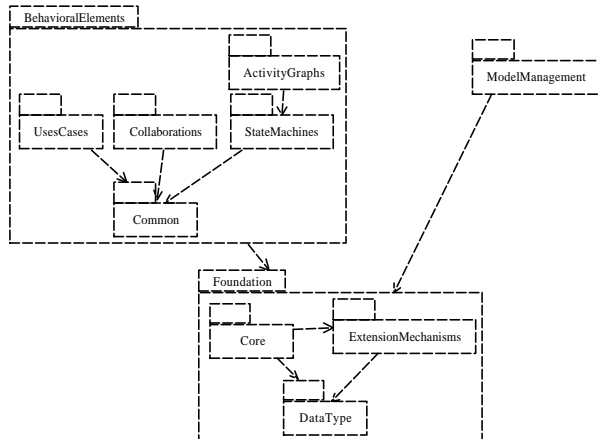
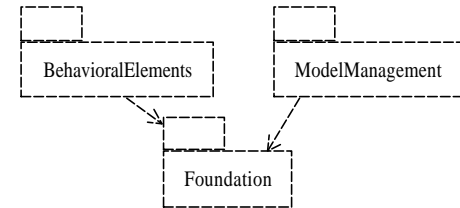
Les noms de paquetage commencent par une lettre majuscule.

Deux éléments à l'intérieur d'un même paquetage ne peuvent avoir le même nom.

Deux éléments à l'intérieur de 2 paquetages différents peuvent avoir le même nom. En fait le nom d'un élément est une abréviation du nom complet: <nomP1> ::<nomP2> ::<nomElement>



Quelques exemples de paquetages (tirés du méta-modèle UML)



Diagrammes d'objets

Conseils

Diagrammes de classes

Ils servent à modéliser l'aspect statique d'un système

On peut les utiliser pour:

- expliquer le **vocabulaire** du système (audit)
- modéliser une collaboration
-

Diagrammes de classes

Utilisation des diagrammes de classes pour un audit (ingénierie du métier, du besoin)

Prendre un mot important du vocabulaire, il correspond à un objet → l'abstraire en classe

Construire un nouveau diagramme de classes

Construire la classe (si elle n'existe pas déjà), la placer au centre du diagramme

Placer autour les mots du vocabulaire (en tant que classes) et relier les classes entre elles avec les relations appropriées.

Recommencer avec un autre mot important du vocabulaire

Dans le cadre d'un projet logiciel, en modélisation métier, il ne doit avoir aucune référence à la future application

Diagrammes de classes

Conseils

Sur le fond

- Un seul thème par diagramme.
Utiliser plusieurs diagrammes !!!
- Il ne doit contenir que des éléments nécessaires
- Les détails doivent être en relation avec le niveau d'abstraction (attribut et opérations des classes, décoration des associations,...)
- Pas plus dépouillé que nécessaire, il ne doit pas induire en erreur le lecteur
- Ne pas être trop précis trop vite!

Sur la forme

- Le nom doit exprimer clairement le thème du diagramme
- Éviter les croisements, rapprocher les éléments liés

Diagrammes d'objets

Exemple

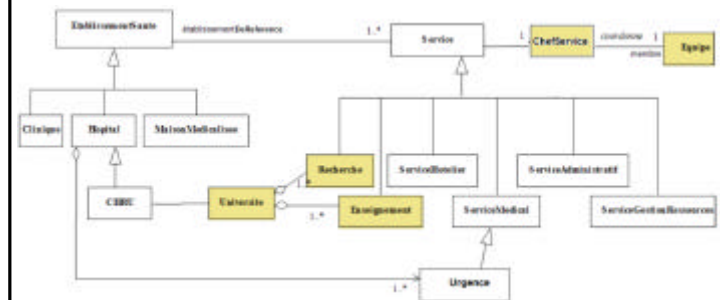
Utilisation de diagrammes de classes pour décrire une structure (modélisation métier)

Un établissement de santé est composé de différents services (médicaux, administratifs, gestions des ressources, hôtelier,...). A la tête de chaque service, on trouve un chef de service qui coordonne une équipe.
Différents types d'établissement de santé, les maisons médicalisées, les cliniques, les hôpitaux qui doivent obligatoirement mettre en place un service des urgences. Parmi les hôpitaux, les CHRU sont en relation avec une université.

Règle métier:

Un chef de service est un médecin hospitalier.

Utilisation de diagrammes de classes pour décrire une structure (modélisation métier)



les agrégats

Il s'agit d'une association par référence, c'est-à-dire que la classe référencée peut avoir une existence en dehors de la classe référençante

Structuration en paquets

L'exemple ci-dessus montre que très rapidement, il est nécessaire de regrouper pour s'y retrouver.

On décide de créer un paquetage organisation, un paquetage LesPersonnes et un paquetage Général où on mettra ce qu'on n'arrive pas à mettre dans les autres paquetages et qui ont un sens en dehors de ces paquetages.

Des dépendances existent entre ces paquetages

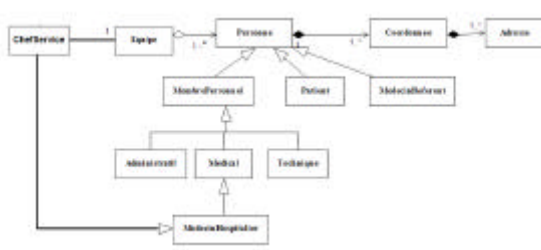
On construit un diagramme de classes santeStructure pour introduire les dépendances:

Utilisation de diagrammes de classes pour décrire une structure (modélisation métier)



Chaque paquetage peut être traité indépendamment (re divisé, nouveaux diagrammes,...)

Le paquetage LesPersonnes

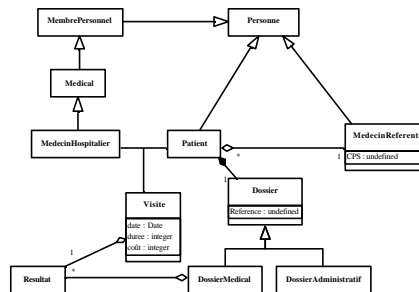
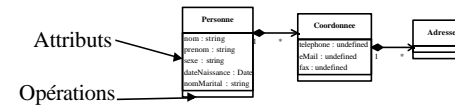


les agrégats forts ou composition.

La classe référencée ne peut pas avoir d'existence en dehors de la classe référençante

Si une instance de Personne est créée, une instance (au moins) de Coordonnée est créée également ; si l'instance de Personne disparaît, l'instance de coordonnée disparaît également. Les durées de vie sont liées

Zoom sur Personne (changement de granularité)



La classe association: Visite

Liens Web

- Cours complet UML2 et modelisation
 - <http://laurent-audibert.developpez.com/Cours-UML/>
- <http://uml.free.fr/>
 - Conception avec UML (en français)