CENTRALE
L Y O N

Approche Objet, UML et ses formalismes

Origines et buts
Principes
Formalismes UML et quelques éléments d'utilisation
AGL supportant UML

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Approche Objet et formalismes UML

Plan

1. Origines et buts
2. Principes
3. Formalismes UML et quelques éléments d'utilisation
4. AGL supportant UML

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les éléments constitutifs d'une démarche de construction de logiciel :

- Buts
- Processus et cycle de vie
- Méthodes
- Formalismes et outils

La représentation triaxiale d'un objet

- Les cas d'utilisation : le savoir-faire
- La description : classes, propriétés, associations
- La dynamique : états, événements

GL/BTD/UML

3

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML (Unified Modeling Language)

Un formalisme issu des méthodes :

- OMT - James Rumbaugh,
- Booch - Grady Booch,
- OOSE - Ivar Jacobson

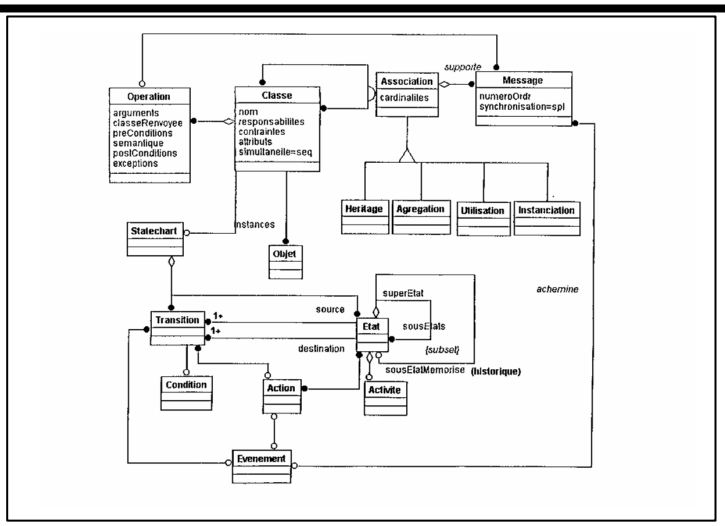
GL/BTD/UML

4

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Méta-modèle de Booch



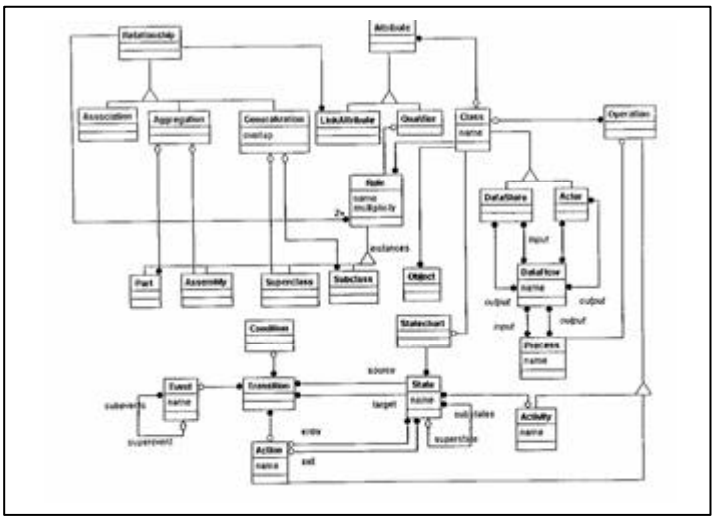
GL/BTD/UML

5

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Méta-modèle de OMT



GL/BTD/UML

6

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Comparaison modèle objet

	OMT	Booch
Classe	✔	✔
abstraite	✔	✔
associative	✔	
générique		✔
Attribut	✔	✔
qualifieur/clé	✔	✔
de lien	✔	
statique / de classe		✔
privé/protégé/public		✔

	OMT	Booch
Association	✔	✔
binaire	✔	✔
n-aire	✔	
rôles	✔	✔
Agrégation	✔	✔
valeur / référence		✔
Généralisation	✔	✔
clusters	✔	
non recouvrement	✔	
Utilisation		✔
Instanciation		✔

GL/BTD/UML 7

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Comparaison modèle dynamique

	OMT	Booch
Opération	✔	✔
arguments	✔	✔
type résultat	✔	
classe renvoyée		✔
pré/post conditions		✔
exceptions		✔
implémentation	✔	✔
Message	✔	✔
Visibilité objets	✔	✔
Scénario	✔	✔
Diag. interaction	✔	✔

	OMT	Booch
Graphe d'états	✔	✔
état, transition	✔	✔
orthogonalité	✔	✔
événement	✔	✔
action, activité	✔	✔
condition	✔	✔
sous-état	✔	✔
historique		✔
Sous-système	✔	✔
Module	✔	✔

GL/BTD/UML 8

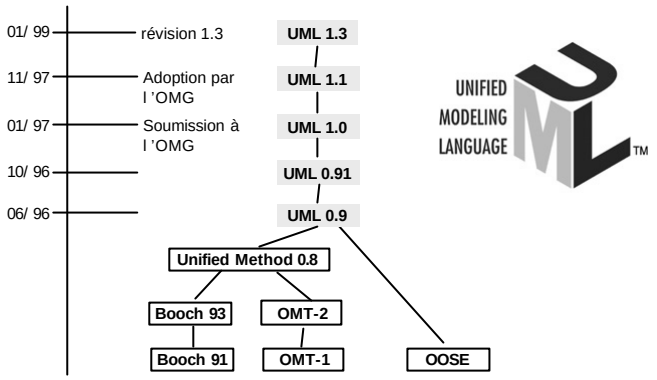
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Historique

UML est un langage de modélisation objet
et non une méthode



GL/BTD/UML

9

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Approche Objet et formalismes UML

Plan

1. Origines et buts
2. Principes
3. Formalismes UML et quelques éléments d'utilisation
4. AGL supportant UML

GL/BTD/UML

10

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Introduction

- UML est un langage formel (ou pseudo-formel) basé sur un métamodèle.
- Le métamodèle permet de définir :
  - les concepts et éléments de modélisation
  - la sémantique de ces éléments
- UML se base sur une notation graphique
- UML propose neuf types de diagrammes
  - représentent les aspects statiques et dynamique
  - couvrent l'ensemble des phases de développement
- UML est ouvert et extensible

GL/BTD/UML 11

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Approche Objet et formalismes UML

Plan

1. Origines et buts
2. Principes
3. Formalismes UML et quelques éléments d'utilisation
4. AGL supportant UML

GL/BTD/UML 12

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

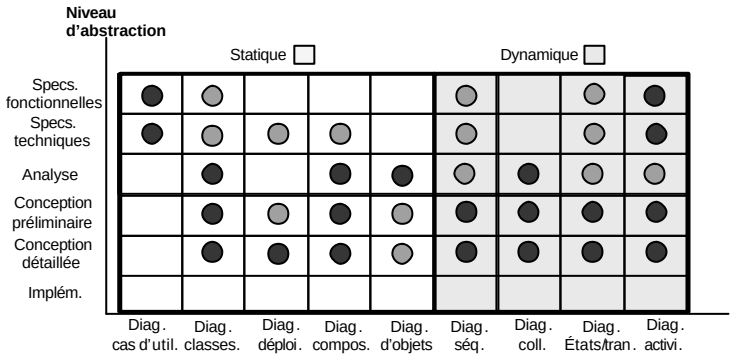
13

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML



GL/BTD/UML

14

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

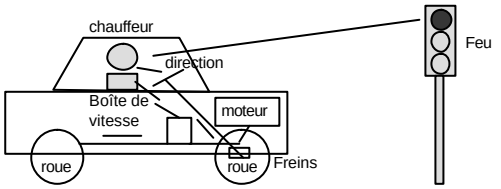
15

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Concepts des Objets

Le monde est composé d'entités qui « collaborent »



- L'approche objet consiste à résoudre un problème en termes d'objets qui collaborent.
- Ces objets sont des abstractions des objets réels

GL/BTD/UML

16

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concepts des Objets

Définition

Un objet est défini par :

- Un état — Ensemble de valeurs associées à des propriétés qui permettent de décrire un objet à un temps t
- Un comportement — Ensemble de services ou d'opérations que peut rendre un objet ou qui modifient son état
- Une identité — Propriété qui permet de distinguer un objet des autres objet

Exemple :

Immat. : *1717 KK 69*
couleur : *bleue*
roues : *4*
puissance : *2CV*
vitesse : *50 km/h*

accélérer
freiner
tourner
reculer

GL/BTD/UML

17

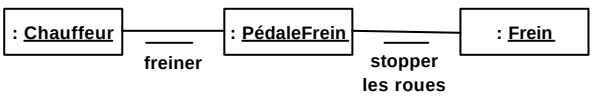
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concepts des Objets

Communication entre objets

Les objets communiquent entre eux par l'envoi de messages



Un message entraîne l'activation d'un ou de plusieurs services de l'objet

GL/BTD/UML

18

CENTRALE
L Y O N

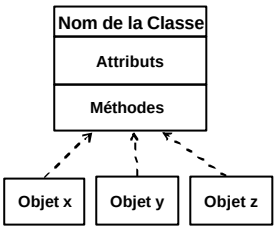
Bertrand DAVID : Génie Logiciel

Concepts des Objets

Classes d'objets

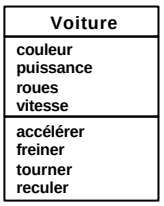
Plusieurs objets possèdent des caractéristiques communes
On regroupe ces caractéristiques dans un même ensemble

La classe d'un objet



Instances de la classe

Exemple :



GL/BTD/UML

19

CENTRALE
L Y O N

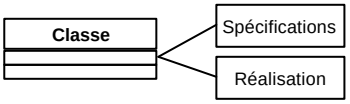
Bertrand DAVID : Génie Logiciel

Concepts des Objets

Concepts fondamentaux

Encapsulation

C'est le fait de « cacher » la réalisation d'une classe
et limiter l'accès à ses propriétés



Intérêts :

- Protéger la façon de réaliser - le code
- Cacher la complexité
- Maintenance facilitée
- Sécurité - filtrer les accès aux données

GL/BTD/UML

20

CENTRALE LYON

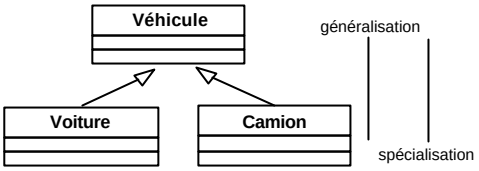
Bertrand DAVID : Génie Logiciel

Concepts des Objets

Concepts fondamentaux

Spécialisation / généralisation

On constate que certaines classes ont des propriétés et des comportements communs qu'il serait intéressant de factoriser



Intérêts :

- Réduire la complexité grâce à la classification / hiérarchisation
- Héritage de propriété = économie de code
- **Réutilisation**

GL/BTD/UML

21

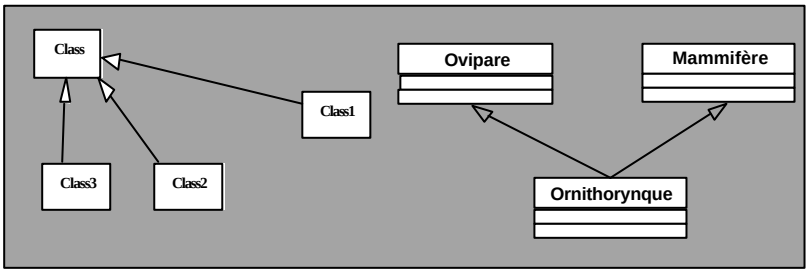
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concepts des Objets

Concepts fondamentaux

Héritage simple ou multiple



A choisir selon le langage d'implémentation qui sera utilisé :

- Java : héritage simple
- C++ : Héritage multiple

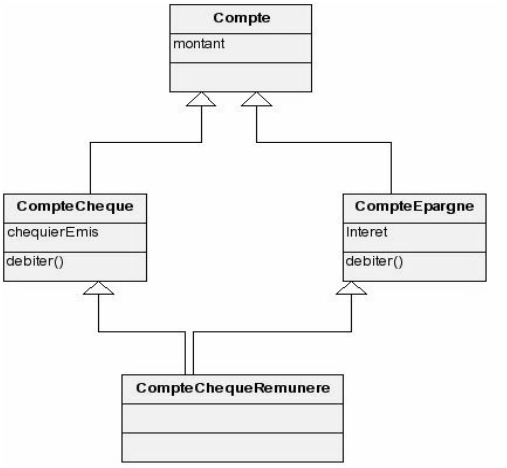
GL/BTD/UML

22

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Héritage en diamant



GL/BTD/UML

23

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

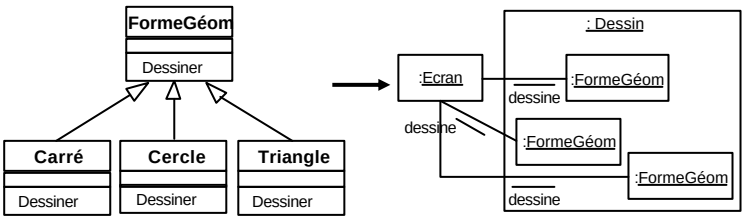
Concepts des Objets

Concepts fondamentaux

Polymorphisme

- C'est un mécanisme qui permet à un objet client d'utiliser les services d'une interface « générique », sans savoir quel est l'objet qui va véritablement répondre à la requête
- Mécanisme qui permet à des objets partageant une interface commune d'en réaliser les opérations de façon propre

Exemple :



GL/BTD/UML

24

CENTRALE
L Y O N

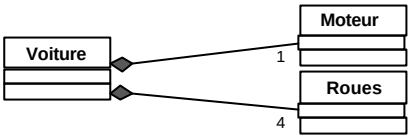
Bertrand DAVID : Génie Logiciel

Concepts des Objets

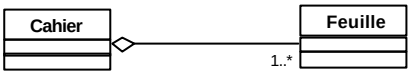
Relations entre classes

- Association : lien sémantique entre deux classes

- Agrégation : relation maître/esclave, contenu/contenant



- Composition : agrégation forte



GL/BTD/UML

25

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

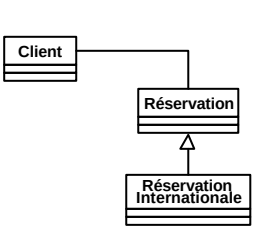
UML - Le langage Objet unifié

Les diagrammes d'UML

Les diagrammes de classes

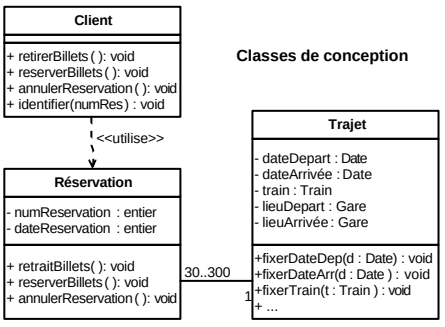
Buts :

1. Structurer l'application - Relations entre classes
2. Base pour générer le code



Classes candidates des cas d'utilisation

(1)



Classes de conception

(2)

GL/BTD/UML

26

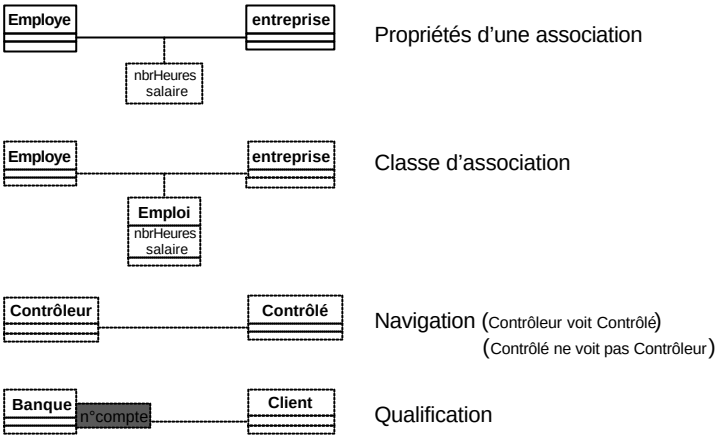
CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML

Les diagrammes de classes - suite



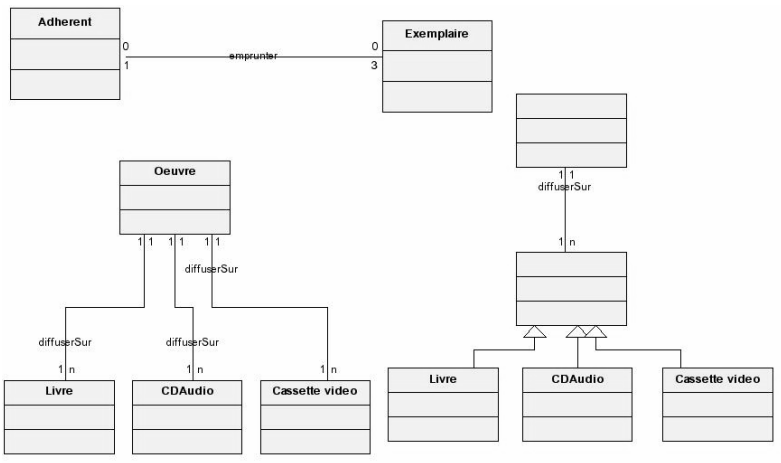
GL/BTD/UML

27

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Diagramme de classes



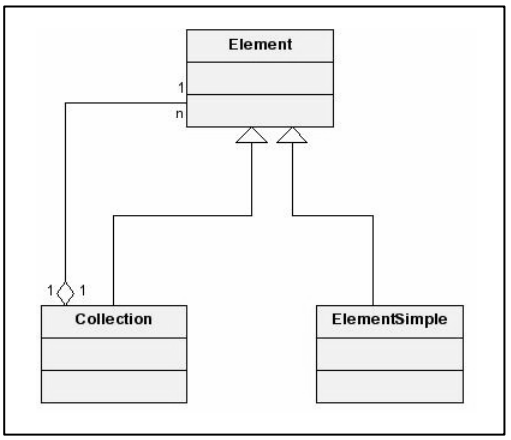
GL/BTD/UML

28

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Agrégation récursive
(pour modéliser des collections emboîtées)



GL/BTD/UML

29

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML - le modèle objet statique

- Classe (nom de la classe, attributs, services)
- Instance d'une classe
- Association (deux classes, nom de l'association, deux noms des rôles, deux cardinalités)
- Classes et Instances
- Cardinalités

GL/BTD/UML

30

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Comment élaborer le diagramme de classes

Pour identifier les classes on peut s'appuyer sur les mots du domaine applicatif

Réduire l'ensemble en fonction des critères suivants :

- supprimer des synonymes,
- supprimer des classes trop vagues,
- supprimer des classes non pertinentes,
- découvrir les associations exprimant l'interdépendance des classes,
- trouver les attribut des classes,
- trouver les opérations sur les classes
- Association ou attribut ?

GL/BTD/UML

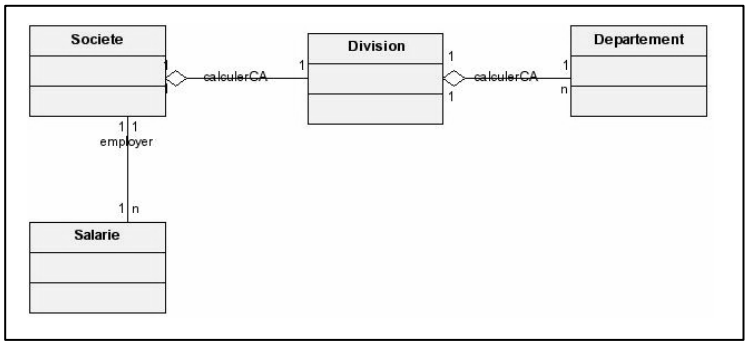
31

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

L'agrégation une association du type "composé-composant"

Les objets agrégés n'ont de raison d'être que s'ils sont liés à l'agrégat.



GL/BTD/UML

32

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Pour s'orienter vers une agrégation :

- l'association a un nom proche de "est composé de" ou "est partie de »,
- il existe une forte différence de granularité entre une première classe (agrégat) et d'autres classes (agrégées),
- la suppression d'un objet agrégat induit la destruction d'autres objets agrégés,
- la modification d'un attribut dans un objet agrégat concerne les attributs des objets agrégés,
- la définition de l'opération d'un objet agrégat repose sur des opérations d'autres objets agrégés. Dans ce cas, les opérations ont souvent des noms similaires.

GL/BTD/UML 33

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Pour améliorer le diagramme de classes :

Inclure la généralisation
Trouver les agrégations
Factoriser avec les interfaces
Utiliser les contraintes

Organiser le diagramme de classes en construisant des packages

Valider le modèle avec les utilisateurs

Incrémenter le modèle (en itérant)

GL/BTD/UML 34

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

35

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

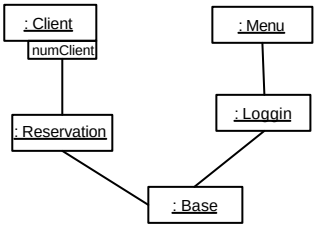
UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes d'objets

Buts :

1. Décrire schématiquement les relations entre objets
2. Support pour la recherche de diagramme de classes et les diagrammes de collaboration



GL/BTD/UML

36

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

37

CENTRALE LYON

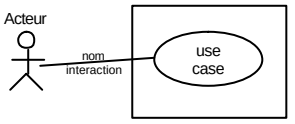
Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML

Les cas d'utilisation

But : spécifications (besoins) fonctionnelles du système



- un cas = un service (fonctionnalité)
- Acteur = utilisateur du service
- Il y a des acteurs principaux et secondaires

Diagramme complété par un document textuel semi-structuré

Titre
But
Résumé
Acteurs
Date + version
Pré conditions
Enchaînements
Exceptions
Post conditions
{IHM}

Les cas d'utilisation sont de très bons moyens de communication

GL/BTD/UML

38

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

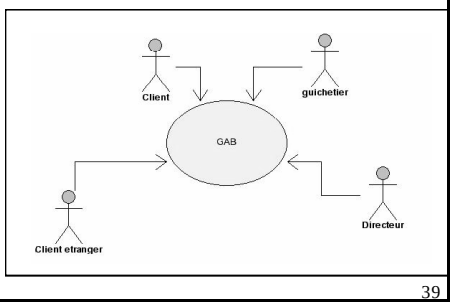
Acteurs

Les acteurs peuvent être de trois types :

- humains, des utilisateurs du logiciel,
- logiciels qui manipulent le systèmes à l'aide d'une API,
- matériels, robots et automates qui exploitent les données du système ou qui sont pilotés par le système.

Typologie des acteurs :

- utilisateurs du système
- administrateurs du système



GL/BTD/UML

39

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Diagrammes de cas d'utilisation

Les cas utilisation expriment

- les principales tâches de chaque acteur,
- les modifications des données du système,
- les cas d'anomalies

Pour voir de façon simple :

- les différents acteurs
- comment est délimité le système
- les fonctionnalités demandées au système
- les rôles des différents acteurs vis-à-vis du système

Descriptions :

- Textuelle
- Sous forme de scénario (diagramme de séquence)
- Sous forme de diagramme de collaboration

GL/BTD/UML

40

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

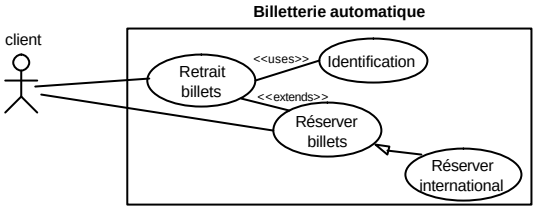
UML - Le langage Objet unifié

Les diagrammes d'UML

Les cas d'utilisation : relations entre cas

Trois types de relations

a - utilisation
b – extension **« extends »** (Pour factoriser et réutiliser).
c - spécialisation / héritage **« uses »** (Pour hériter et affiner)



Attention au piège de la décomposition trop fine : fonctionnelle

GL/BTD/UML

41

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les diagrammes dynamiques

- Scénario – Diagramme de séquence
- Diagramme d'états
- Diagramme d'activités
- Diagramme de collaboration

GL/BTD/UML

42

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

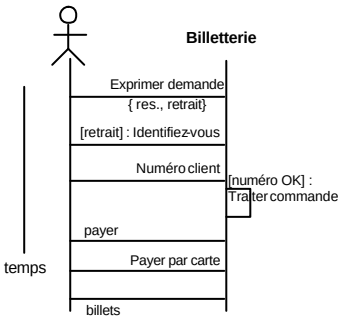
UML - Le langage Objet unifié

Les diagrammes d'UML

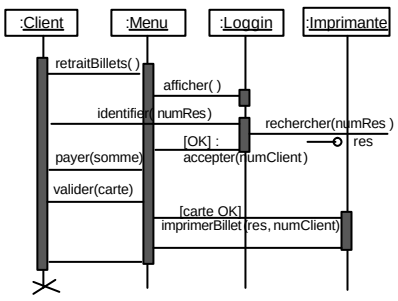
Les diagrammes de séquence

Buts :

1. décrire les cas d'utilisation (scénarios)
2. décrire les interactions entre objets



(1)



(2)

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Scénario : Diagramme de séquence

Un **scénario** est une série d'événements ordonnés dans le temps, simulant une exécution particulière du système.

Les scénarios permettent d'expérimenter les exécutions du système, ils sont donc très utiles pour les phases de tests et de maintenance.

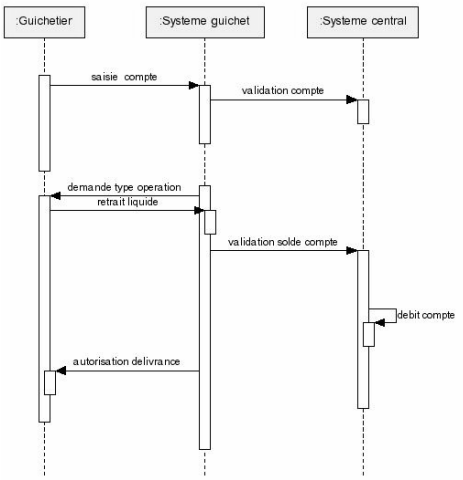
GL/BTD/UML

45

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Diagramme de séquence



GL/BTD/UML

46

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

47

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

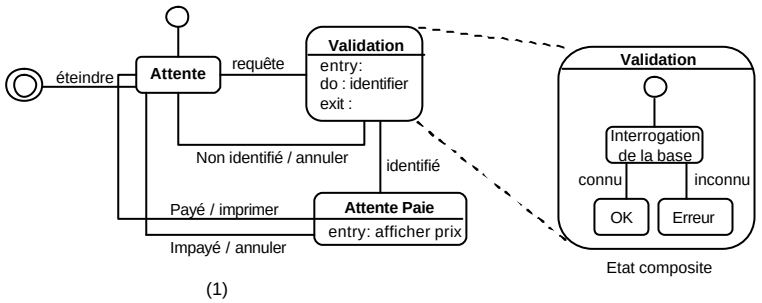
UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes d'états / transitions

Buts :

1. Illustrer les cas d'utilisation
2. Décrire en détail le comportement des classes



(1)

GL/BTD/UML

48

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Le modèle dynamique

Pourquoi un modèle dynamique ?

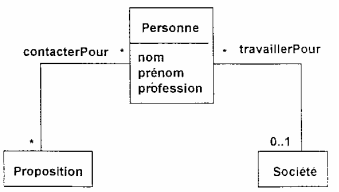
- pour décrire les relations temporelles et événementielles,
- pour exprimer les états en fonction de modifications internes et d'options choisies par les utilisateurs,
- pour indiquer les actions possibles dans un contexte donné pour décrire les actions des systèmes extérieurs sur les objets du système étudié ainsi que les réactions de ces objets

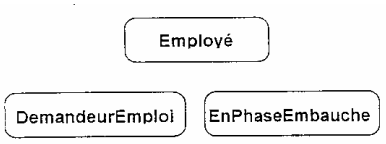
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concepts et notations de base (1)
La notion d'état

- L'état d'un objet est défini à la fois par les valeurs de ses variables d'instance et par les valeurs de ses liens avec d'autres objets. L'état d'un objet correspond à une durée, un intervalle de temps quantifiable à l'échelle du système.





CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Concepts et notations de base (2)
La notion d'événement

- Un **événement** est un stimulus pouvant transporter des informations (sous forme de paramètres), il se produit à un instant donné : un événement n'a pas de durée contrairement à un état.
- Un événement peut être émis par un objet du système ou par un objet externe au système, cela correspond à l'appel d'une méthode. Un événement peut également provenir de l'interface du système (clic souris, sélection d'un item dans un menu,…).
- La réponse d'un objet à un événement dépend de l'état dans lequel se trouve l'objet qui le reçoit.

GL/BTD/UML 51

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Concepts et notations de base (3)
La notion de message

- Un **message** est un événement particulier, issu de l'interaction entre deux objets, un objet appelle une méthode d'un autre objet.
- Tout message est un événement impliqué dans l'interaction entre deux objets.
- Tout événement n'est pas un message, car il n'est pas forcément émis par un objet.

GL/BTD/UML 52

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

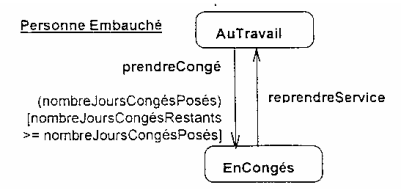
Diagramme d'états

Un **diagramme d'états** est un graphe constitué de noeuds représentant des états ainsi que des flèches représentant des transitions portant des paramètres et des noms d'événements.

Un diagramme d'états est propre à une classe donnée :

Un diagramme d'états ne permet pas de représenter les relations d'interaction entre les objets intervenant dans un système, mais le cycle de vie des objets d'une classe.

Un état peut avoir des sous-états

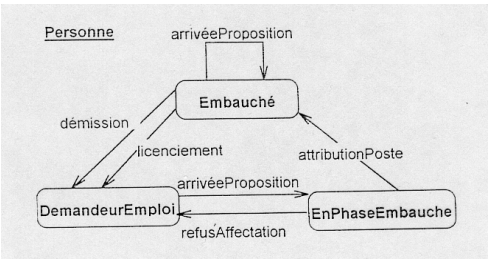


CENTRALE LYON

Bertrand DAVID : Génie Logiciel

La notion de transition

Une **transition** est le changement d'état d'un objet causé par un événement. Les transitions sont représentées par des flèches portant le nom et les paramètres des événements associés.



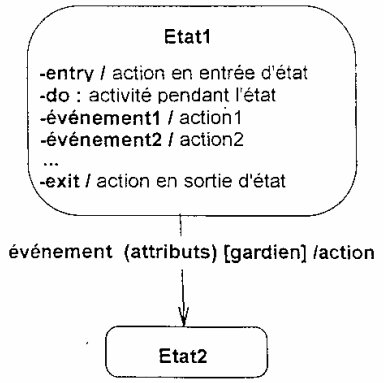
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Caractérisation d'une transition (1)

Les **attributs** qui correspondent à des informations ou des paramètres portés par des événements.
Les **gardiens** ou conditions qui sont des fonctions booléennes.
Deux types d'opérations peuvent être impliquées dans un diagramme dynamique : les activités et les actions

- Une **activité** est une opération continue dans le temps, elle prend un certain temps pour se réaliser. Elle est forcément associée à un état.
- Une **action** est une opérations instantanée, elle est réalisée de façon immédiate, et peut être associée aussi bien à l'état d'un objet qu'à une transition. Elle peut intervenir soit en entrée d'état (préfixe entre/), soit en sortie d'état (préfixe exit/), soit en réponse à un événement déclencheur (préfixe NomEvenement/), soit enfin en cours d'une transition.



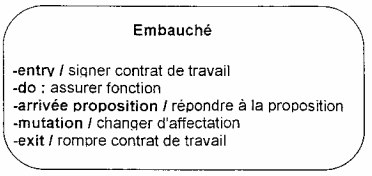
GL/BTD/UML

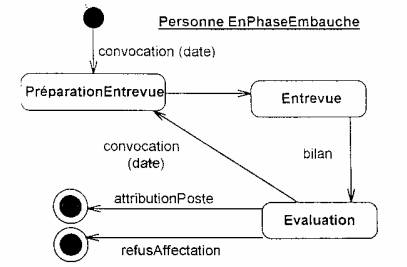
55

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Caractérisation d'une transition (2)





GL/BTD/UML

56

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concurrence et synchronisation d'états

Plusieurs sous-diagrammes d'états peuvent intervenir en parallèle dans un même état. Ils sont alors :

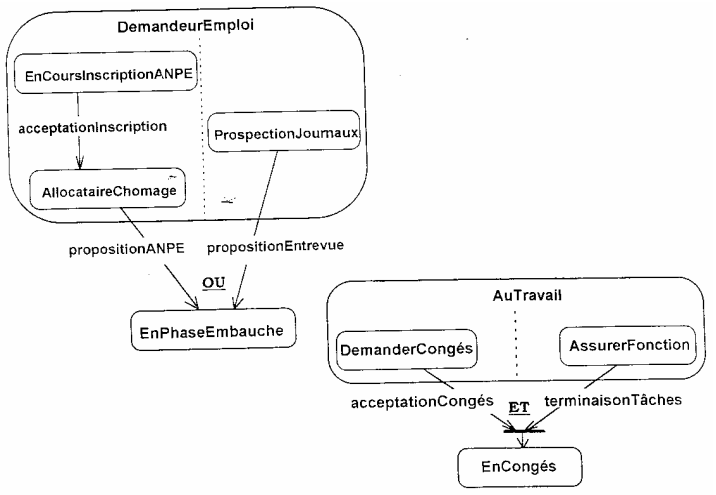
- soit en concurrence : le premier sous-diagramme permettant de faire la transition de l'état englobant vers un autre état interrompt les autres sous-digrammes et fait quitter l'état englobant.
- soit en synchronisation : la transition de l'état englobant vers un autre état n'est effectuée que lorsque tous les sous-diagrammes le permettent, aucun sous-diagramme ne pouvant être interrompu.

GL/BTD/UML 57

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concurrence et synchronisation d'états



GL/BTD/UML 58

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

59

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

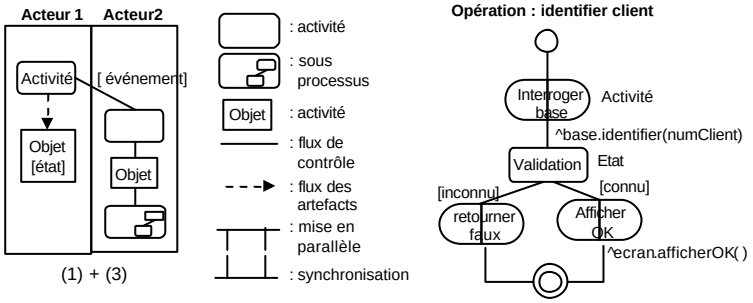
Les diagrammes d'UML

Diagrammes d'activités

Buts :

1. Décrire le comportement générique d'un use case
2. Décrire en détail le comportement d'une opération
3. Modéliser les processus métiers

Dual des diagrammes d'états / transitions



GL/BTD/UML

60

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

61

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

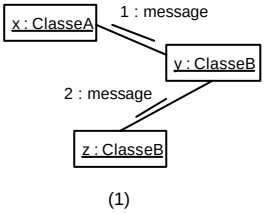
UML - Le langage Objet unifié

Les diagrammes d'UML

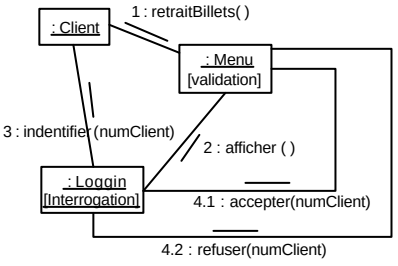
Diagrammes de collaboration

Buts :

1. Décrire l'interaction des objets entre eux
2. Illustrer les scénarios des use cases
3. Valider les choix d'analyse et de conception (prototypage)
4. Aider à élaborer des diagrammes de classes de conception



(1)



(1) + (3)

GL/BTD/UML

62

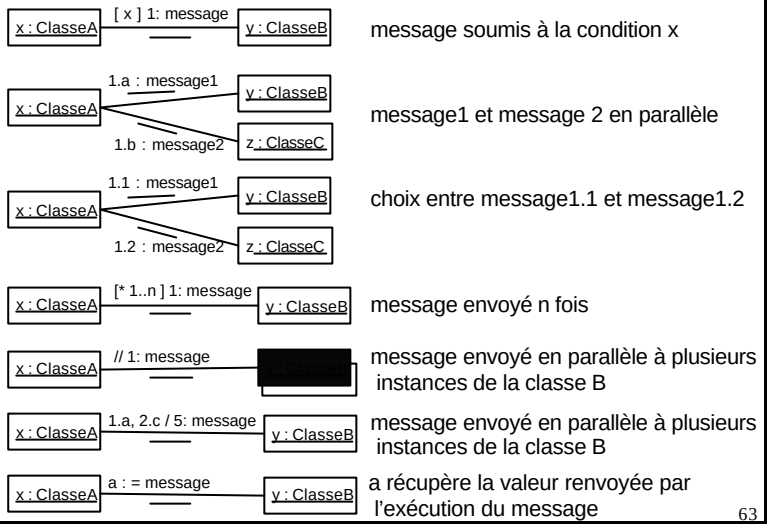
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes de collaboration - conventions



CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Diagramme de collaboration entre objets

Un **diagramme** de collaboration entre objets vise à représenter du point de vue statique et dynamique les objets impliqués dans la mise en place d'une fonction applicative.

L'objectif est de construire un modèle expliquant la coopération entre les objets utilisés pour la réalisation d'une fonctionnalité.

Deux notions fondamentales :

- Le **contexte** est une vue statique partielle des objets qui collaborent pour réaliser une fonction.
- Les **interactions** entre objets décrites dans un diagramme de collaboration sont des séquences de messages échangés par les objets dans le cadre de la réalisation d'une opération.

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes de composants

Buts :

1. Structurer l'application - Architectures (fonctionnelle/logicielle)
2. Regrouper des éléments à forte cohérence dans des « conteneurs » faiblement couplés (encapsulation de haut niveau)
3. Faciliter la maintenance (évolution - adaptation - debug)
4. Construction de frameworks (réutilisation - « off the shelf »)

GL/BTD/UML 65

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML 66

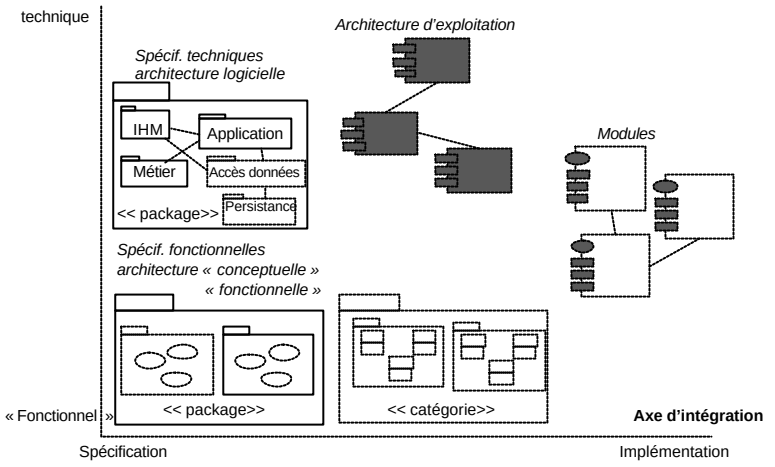
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes de composants



GL/BTD/UML

67

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML et ses formalismes

- Diagramme de classes
- Diagramme d'objets
- Diagramme de cas d'utilisation
- Diagramme d'états
- Diagramme de séquences
- Diagramme d'activités
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement

GL/BTD/UML

68

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

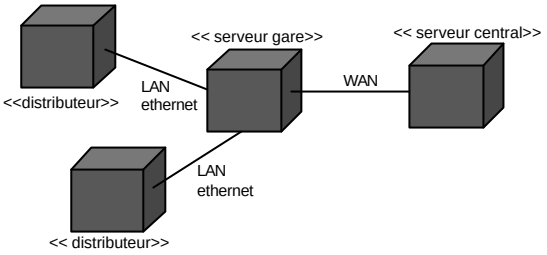
UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes de déploiement

But :

Décrire l'architecture matérielle



GL/BTD/UML

69

CENTRALE LYON

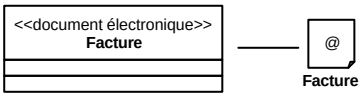
Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Les diagrammes d'UML

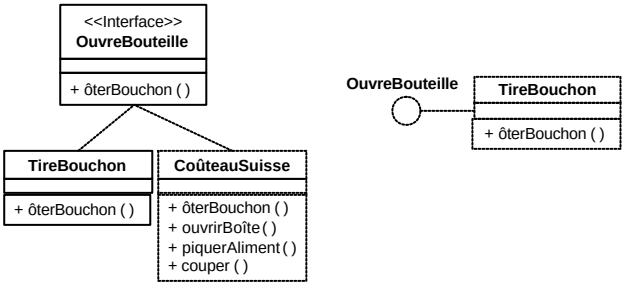
Autres concepts

- Stéréotype : mécanisme d'extension d 'UML



Induit certaines propriétés et certains comportement spécifiques

- Interface : « façade » - ensemble de services



GL/BTD/UML

70

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concepts des Objets

Concepts complémentaires

- **Classe abstraite**
  - Classe très générale, non instanciable
  - Sert à la classification / hiérarchisation
- **Classe générique**
  - Classe paramétrable dont les comportements peuvent être utilisés pour des types différents
- **Métaclasse**
  - Classe dont les instances sont des classes

GL/BTD/UML

71

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les interfaces

- Une interface est une classe particulière qui ne contient que des opérations. Elle ne présuppose rien de l'implémentation de ces opérations, mais spécifie leur sémantique.
- Une interface permet donc de décrire certaines caractéristiques de classe en dehors de toute hiérarchie.

GL/BTD/UML

72

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

L'interface *Comparable*

- L'interface *Comparable* définit l'ensemble des opérations qui permettent de comparer deux objets. On dit qu'une classe qui implémente l'ensemble de ces opérations *implémente* l'interface.
- Une interface peut être vue comme une classe. Sa représentation est exactement la même. Il suffit de rajouter le mot-clé " interface " au-dessus du nom de la classe. Il existe toutefois une représentation plus simple qui peut être utilisée lors de l'implémentation.
- UML permet la création de nouvelles interfaces en les faisant hériter d'interfaces existantes. L'héritage consiste simplement à reprendre les opérations de la classe mère dans les spécifications de la classe fille.

GL/BTD/UML 73

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Interface ou généralisation ?

- Les interfaces définissent souvent des comportements génériques qui ne peuvent pas être encapsulés dans des classes mères car ils ne font pas partie de la sémantique propre aux objets.
- Les noms des interfaces se terminent souvent par le suffixe " able ", ce qui signifie que des interfaces représentent seulement certaines aptitudes de classes.
- Ces aptitudes caractérisent des potentialités de classes qui ne font pas partie des caractéristiques intrinsèques de la classe.

GL/BTD/UML 74

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Exemple d ’interface :

- Pour comparer des clients par le montant total de leurs achats il est plus judicieux de faire hériter *Client* de *Personne* et de lui faire implémenter une interface *Comparable* que de faire hériter *Client* d’une classe *Comparable*.

GL/BTD/UML

75

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Les packages

Un package regroupe en ensemble d'entités qui correspondent à une fonctionnalité bien définie.

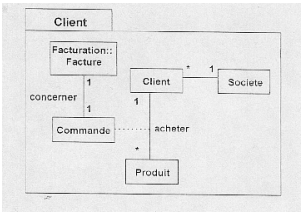
Le package est un espace de nommage.
nomDuPackage :: NomDeLaClasse

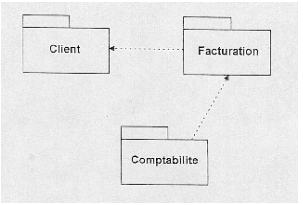
Deux présentations :

- explicite montrant le contenu du package
- limitée à la vision globale (sans montrer l'intérieur)

Les interactions entre packages :

- mécanisme de dépendance - toute modification du package source entraîne des modifications du package pointé.





GL/BTD/UML

76

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Concepts des Objets

Concepts complémentaires

- **Catégories**
  - Regroupement de classes à forte cohérence internes
  - Très faible couplage avec les catégories extérieures
- **Composants**
  - Regroupement de classes perçues comme une boîte noire et proposant un ensemble bien défini de services
- **Frameworks**
  - Ensemble de classes proposant une architecture
  - Frameworks métiers et techniques

Intérêts :
- Maîtrise de la complexité
- Réutilisation

GL/BTD/UML 77

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML est un modèle ouvert : on peut introduire des nouveaux concepts

Le concept de stéréotype

Les stéréotypes permettent de créer de nouveaux concepts au sein du modèle.

Exemple : Le stéréotype “ persistance ” indique que la classe *Client* est capable de se connecter à une base de données

GL/BTD/UML 78

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les stéréotypes

- ils classifient des éléments du modèle. Ils permettent donc de créer des familles d'éléments associés à un même stéréotype ;
- ils modifient la sémantique des éléments associés et permettent la création de nouveaux concepts propres à une application.

  Attention toutefois à l'intégrité du métamodèle de UML.

GL/BTD/UML 79

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les contraintes

La contrainte permet de préciser le contexte du modèle en positionnant des restrictions.

GL/BTD/UML 80

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les qualificateurs

- Pour réduire une cardinalité à 1
- Une partition est la décomposition d'un ensemble E en sous-ensembles disjoints dont la réunion forme l'intégralité de l'ensemble E.
- Un dictionnaire est une collection d'éléments, chaque élément étant une association entre une clé et une valeur.
- Une clé primaire est un identifiant pour un enregistrement d'une base de données.

GL/BTD/UML

81

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

UML - Le langage Objet unifié

Avantages / Inconvénients

- 1. Langage formel
- 2. Langage standardisé et normalisé → très répandu
- 3. De nombreux AGL
- 4. Formalismes graphiques
- 5. Plusieurs types de diagrammes - différents niveaux et vues
- 6. Forte cohérence entre diagrammes
- 7. Les diagrammes sont issus de formalismes existants
- 8. Extensibilité
- 2 + 7 : **moyen de communication entre équipes**

- 1. Long à maîtriser
- 2. Pas de méthode

GL/BTD/UML

82

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Appliquer UML

Où	Quoi
Développement Logiciel	Tout
Analyse des besoins fonctionnels	Use case, Diag. Séquences
Elaboration du cahier des charges	Diag. Activités
Modélisation de processus	Diag. d'activités, Diag. de classes
Temps Réel	Diag. D'états/transitions, de classes
	Nouvelles spécs . UML (1.4)
Modélisation de collecticiels	Diag. De collaboration, de classes

GL/BTD/UML

83

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Approche Objet et formalismes UML

Plan

1. Origines et buts
2. Principes
3. Formalismes UML et quelques éléments d'utilisation
4. AGL supportant UML

GL/BTD/UML

84

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

AGL UML

Critères de sélection

- Nombre de diagrammes supportés
- Génération automatique de diagrammes
- Génération de code - langages supportés
- Rétro ingénierie (reverse engineering)
- Prototypage
- Synchronisation du code avec modifications extérieures
- API
- Echanges avec d'autres AGL UML ou IDE
- Utilisation pratique (schémas de qualité, convivialité, générer des images, ...)
- Patterns de conception - et création de patterns
- …

GL/BTD/UML 85

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

AGL UML

Quelques AGL :

- Together (Togethersoft) http://www.togethersoft.com
- Rational ROSE (Rational corp.) http://www.rational.com
- Paradigm (Platinium) http://www.platinium .com
- Magic Draw UML (NoMagic) http://www.nomagic.com
- DOM (ObjetDirect) http://www.objetdirect.com
- Objecteering (SoftTeam) http://www.objecteering.com
- Aonix Life Cycle Desktop (Aonix) http://www.aonix.com, .fr
- UML Studio (Pragsoft) http://www.pragsoft.com

GL/BTD/UML 86