

CENTRALE  
L Y O N

## Développement de logiciels basé sur des processus

Approche Objet, UML et ses formalismes

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

BTD/DLBP/UML 1

CENTRALE  
L Y O N

## Approche Objet et formalismes UML

Bertrand DAVID : Développement de logiciels  
basé sur des processus

### Plan

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

BTD/DLBP/UML 2

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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

BTD/DLBP/UML

3

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## UML (Unified Modeling Language)

**Un formalisme issu des méthodes :**

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

BTD/DLBP/UML

4

CENTRALE  
L Y O N

## UML - Le langage Objet unifié

Historique

---

UML est un langage de modélisation objet et non une méthode

01/ 99 ——— révision 1.3

11/ 97 ——— Adoption par l'OMG

01/ 97 ——— Soumission à l'OMG

10/ 96 ———

06/ 96 ———

```

graph TD
    Booch91[Booch 91] --> Booch93[Booch 93]
    OMT1[OMT-1] --> OMT2[OMT-2]
    Booch93 --> UM08[Unified Method 0.8]
    OMT2 --> UM08
    UM08 --> UML09[UML 0.9]
    OOSE[OOSE] --> UML09
    UML09 --> UML091[UML 0.91]
    UML091 --> UML10[UML 1.0]
    UML10 --> UML11[UML 1.1]
    UML11 --> UML13[UML 1.3]
        
```

BTD/DLBP/UML

5

CENTRALE  
L Y O N

## Approche Objet et formalismes UML

---

### Plan


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

BTD/DLBP/UML

6

CENTRALE  
L Y O N

UML - Le langage Objet unifié



Introduction

Bertrand DAVID : Développement de logiciels basé sur des processus

- ▶ 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

BTD/DLBP/UML

7

CENTRALE  
L Y O N

Approche Objet et formalismes UML

Bertrand DAVID : Développement de logiciels basé sur des processus

**Plan**

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

BTD/DLBP/UML

8

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

9

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## UML - Le langage Objet unifié

↳ Les diagrammes d'UML

---

**Niveau d'abstraction**

	Statique <input type="checkbox"/>					Dynamique <input type="checkbox"/>			
Specs. fonctionnelles	●	●				●		●	●
Specs. techniques	●	●	●	●		●		●	●
Analyse		●		●	●	●	●	●	●
Conception préliminaire		●	●	●	●	●	●	●	●
Conception détaillée		●	●	●	●	●	●	●	●
Implém.									
	Diag. cas d'util.	Diag. classes.	Diag. déploi.	Diag. compos.	Diag. d'objets	Diag. séq.	Diag. coll.	Diag. États/tran.	Diag. activi.

10

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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

BTD/DLBP/UML 11

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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

BTD/DLBP/UML 12

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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



13

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## Concepts des Objets

### Communication entre objets

---

Les objets communiquent entre eux par l'envoi de messages

```

graph LR
    A[": Chauffeur"] -- freiner --> B[": PédaleFrein"]
    B -- "stopper les roues" --> C[": Frein"]
    
```

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

14

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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**

Nom de la Classe
Attributs
Méthodes

Objet x
---------

Objet y
---------

Objet z
---------

Instances de la classe

Exemple :

<b>Voiture</b>
couleur
puissance
roues
vitesse
accélérer
freiner
tourner
reculer

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## Concepts des Objets

### Concepts fondamentaux

#### Encapsulation

---

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

Classe
--------

Spécifications
Réalisation

**Intérêts :**

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

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

```

classDiagram
    Véhicule <|-- Voiture
    Véhicule <|-- Camion
    
```

**Intérêts :**

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

17

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## Concepts des Objets

Concepts fondamentaux

Héritage simple ou multiple

---

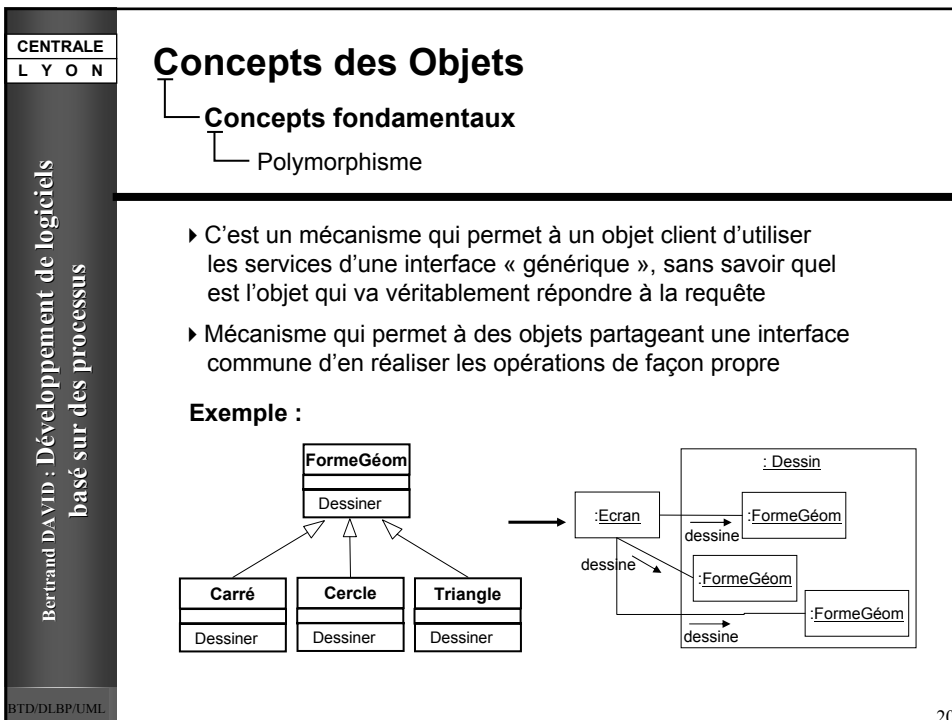
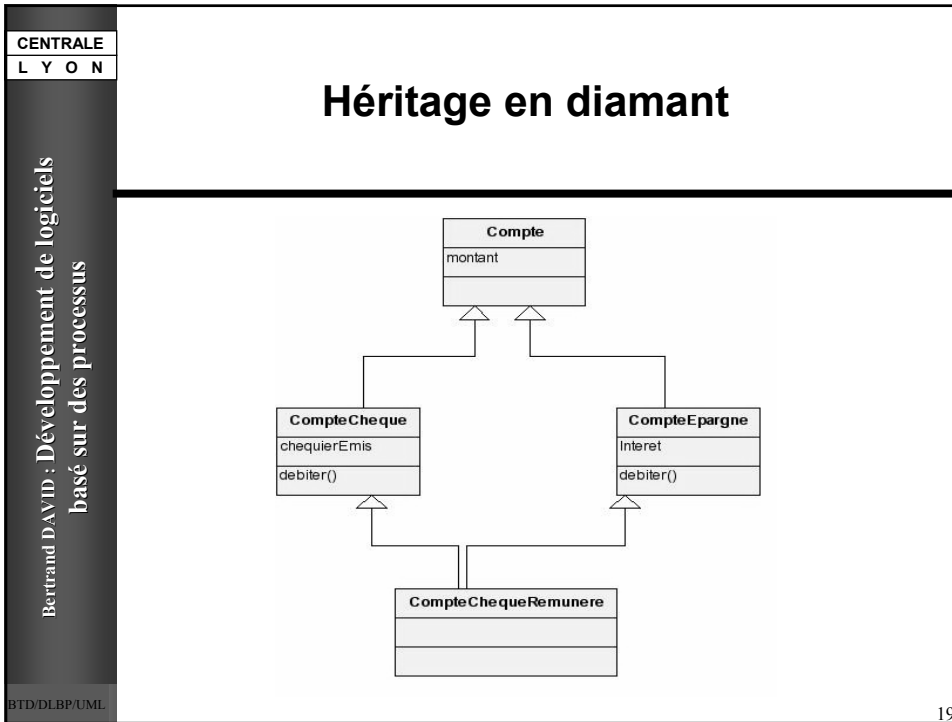
```

classDiagram
    Class <|-- Class1
    Class <|-- Class2
    Class <|-- Class3
    Class <|-- Ovipare
    Class <|-- Mammifère
    Ovipare <|-- Ornithorynque
    Mammifère <|-- Ornithorynque
    
```

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

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

18



CENTRALE LYON

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## Concepts des Objets

### Relations entre classes

---

- ▶ Association : lien sémantique entre deux classes

```

classDiagram
    class Voiture
    class Personne
    Voiture "1..*" -- "1..1" Personne : propriétaire
    
```

- ▶ Agrégation : relation maître/esclave, contenu/contenant

```

classDiagram
    class Voiture
    class Moteur
    class Roues
    Voiture o-- "1" Moteur
    Voiture o-- "4" Roues
    
```

- ▶ Composition : agrégation forte

```

classDiagram
    class Cahier
    class Feuille
    Cahier *-- "1..*" Feuille
    
```

BTD/DLBP/UML

21

CENTRALE LYON

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## 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)

BTD/DLBP/UML

22

CENTRALE LYON

Bertrand DAVID : Développement de logiciels basé sur des processus

## UML - Le langage Objet unifié

UNITED MODELING LANGUAGE

### Les diagrammes d'UML

Les diagrammes de classes - suite

The diagrams show: 1) An association between 'Employe' and 'entreprise' with a property box containing 'nbrHeures' and 'salaire'. 2) An association between 'Employe' and 'entreprise' with an association class 'Emploi' containing 'nbrHeures' and 'salaire'. 3) A directed association from 'Contrôleur' to 'Contrôlé'. 4) An association between 'Banque' and 'Client' with a qualification box containing 'n°compte'.

Propriétés d'une association

Classe d'association

Navigation (Contrôleur voit Contrôlé)  
(Contrôlé ne voit pas Contrôleur)

Qualification

BTD/DLBP/UML 23

CENTRALE LYON

Bertrand DAVID : Développement de logiciels basé sur des processus

## Diagramme de classes

The diagram shows classes: 'Adherent', 'Exemplaire', 'Oeuvre', 'Livre', 'CDAudio', 'Cassette video', and an unnamed class. 'Adherent' (0 to 1) is associated with 'Exemplaire' (0 to 3) via 'emprunter'. 'Oeuvre' (1 to 1) is associated with 'Livre' (1 to n), 'CDAudio' (1 to n), and 'Cassette video' (1 to n) via 'diffuserSur'. An unnamed class (1 to 1) is associated with 'CDAudio' (1 to n) via 'diffuserSur'. 'Livre', 'CDAudio', and 'Cassette video' inherit from the unnamed class.

BTD/DLBP/UML 24

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## Agrégation récursive (pour modéliser des collections emboîtées)

```

classDiagram
    class Element
    class Collection
    class ElementSimple
    Collection "1" o-- "1" Collection
    Collection "1" -- "n" Element
    Element <|-- ElementSimple
    
```

BTD/DLBP/UML

25

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

BTD/DLBP/UML

26

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## Comment élaborer le diagramme de classes

Pour identifier les classes 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 ?

BTD/DLBP/UML

27

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## 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.

```

classDiagram
    class Societe
    class Division
    class Departement
    class Salarie
    Societe "1" o-- "1" Division : calculerCA
    Societe "1" o-- "n" Salarie : employer
    Division "1" o-- "n" Departement : calculerCA
  
```

BTD/DLBP/UML

28

CENTRALE L Y O N	<h2>Pour s'orienter vers une agrégation :</h2>
Bertrand DAVID : Développement de logiciels basé sur des processus	<ul style="list-style-type: none"><li>• l'association a un nom proche de "est composé de" ou "est partie de »,</li><li>• il existe une forte différence de granularité entre une première classe (agrégat) et d'autres classes (agrégées),</li><li>• la suppression d'un objet agrégat induit la destruction d'autres objets agrégés,</li><li>• la modification d'un attribut dans un objet agrégat concerne les attributs des objets agrégés,</li><li>• 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.</li></ul>
BTD/DLBP/UML	29

CENTRALE L Y O N	<h2>Pour améliorer le diagramme de classes :</h2>
Bertrand DAVID : Développement de logiciels basé sur des processus	<p>Inclure la généralisation Trouver les agrégations Factoriser avec les interfaces Utiliser les contraintes</p> <p>Organiser le diagramme de classes en construisant des packages</p> <p>Valider le modèle avec les utilisateurs</p> <p>Incrémenter le modèle (en itérant)</p>
BTD/DLBP/UML	30

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## UML et ses formalismes

31

---

- 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 : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## UML - Le langage Objet unifié

└─ Les diagrammes d'UML

    └─ Diagrammes d'objets

32

---

**Buts :**

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

```

classDiagram
    class Client {
        numClient
    }
    class Menu
    class Reservation
    class Login
    class Base
    Client --> Reservation
    Menu --> Login
    Reservation --|> Base
    Login --|> Base
    
```

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

33

CENTRALE  
L Y O N


Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

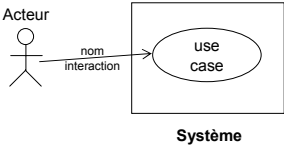
## UML - Le langage Objet unifié

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

34

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

```

graph TD
    Client((Client)) --> GAB((GAB))
    guichetier((guichetier)) --> GAB
    ClientEtranger((Client étranger)) --> GAB
    Directeur((Directeur)) --> GAB
    
```

35

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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


36

CENTRALE  
L Y O N

# UML - Le langage Objet unifié

Les diagrammes d'UML

Les cas d'utilisation : relations entre cas

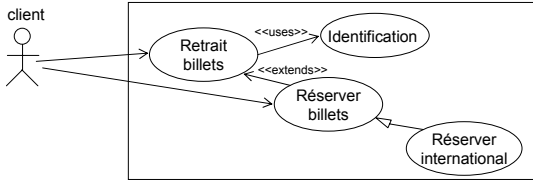


Bertrand DAVID : Développement de logiciels basé sur des processus

Trois types de relations

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

**Billetterie automatique**



Attention au piège de la décomposition trop fine : fonctionnelle

BTD/DLBP/UML

37

CENTRALE  
L Y O N

# Les diagrammes dynamiques

Bertrand DAVID : Développement de logiciels basé sur des processus

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

BTD/DLBP/UML

38

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## 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

39

CENTRALE  
L Y O N


Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## UML - Le langage Objet unifié

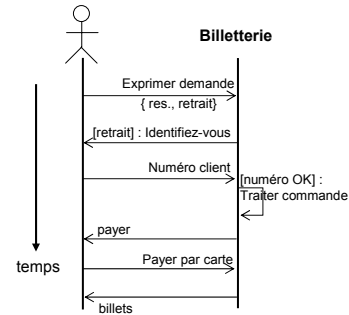
Les diagrammes d'UML

Les diagrammes de séquence

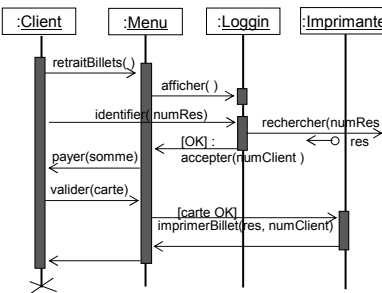


**Buts :**

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



(1)



(2)

40

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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.

41

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## Diagramme de séquence

---

```

sequenceDiagram
    participant G as :Guicheter
    participant SG as :Systeme guichet
    participant SC as :Systeme central

    G->>SG: saisie compte
    activate SG
    SG->>SC: validation compte
    activate SC
    SC->>SG: 
    deactivate SC
    SG->>G: demande type operation
    activate G
    G->>SG: retrait liquide
    activate SG
    SG->>SC: validation solde compte
    activate SC
    SC->>SG: debit compte
    deactivate SC
    SG->>G: autorisation delivrance
    deactivate SG
    deactivate G
    
```

42

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

43

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

```

stateDiagram-v2
    [*] --> Attente
    Attente --> Validation : requête
    Validation --> Attente : Non identifié / annuler
    Validation --> Attente : identifié
    Attente --> AttentePaie : Payé / imprimer
    AttentePaie --> Attente : Impayé / annuler
    Attente --> [*] : éteindre
    state ValidationComposite as Validation
    state ValidationComposite {
        [*] --> Interrogation
        Interrogation --> OK : connu
        Interrogation --> Erreur : inconnu
    }
    ValidationComposite --> Validation : 
    
```

(1)

Etat composite

44

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

BTD/DLBP/UML

45

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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.

```

classDiagram
    class Personne {
        nom
        prénom
        profession
    }
    class Proposition
    class Société
    Personne "1" -- "*" Proposition : contacterPour
    Personne "1" -- "0..1" Société : travaillerPour
    
```

Employé

DemandeurEmploi    EnPhaseEmbauche

BTD/DLBP/UML

46

CENTRALE L Y O N	<h2>Concepts et notations de base (2)</h2> <h3>La notion d'événement</h3>
Bertrand DAVID : Développement de logiciels basé sur des processus	<ul style="list-style-type: none"><li>• Un <b>événement</b> 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.</li><li>• 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,...).</li><li>• La réponse d'un objet à un événement dépend de l'état dans lequel se trouve l'objet qui le reçoit.</li></ul>
BTD/DLBP/UML	47

CENTRALE L Y O N	<h2>Concepts et notations de base (3)</h2> <h3>La notion de message</h3>
Bertrand DAVID : Développement de logiciels basé sur des processus	<ul style="list-style-type: none"><li>• Un <b>message</b> est un événement particulier, issu de l'interaction entre deux objets, un objet appelle une méthode d'un autre objet.</li><li>• Tout message est un événement impliqué dans l'interaction entre deux objets.</li><li>• Tout événement n'est pas un message, car il n'est pas forcément émis par un objet.</li></ul>
BTD/DLBP/UML	48

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## 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

```

stateDiagram-v2
    state "Personne Embauché" as PE
    state AuTravail
    state EnCongés
    PE --> AuTravail : prendreCongé
    AuTravail --> PE : repandreService
    PE --> EnCongés : prendreCongé
    EnCongés --> PE : repandreService
    
```

49

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## 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.

```

stateDiagram-v2
    state "Personne" as P
    state "Embauché" as E
    state "DemandeurEmploi" as DE
    state "EnPhaseEmbauche" as EP
    P --> E : arrivéeProposition
    E --> DE : démission
    E --> DE : licenciement
    DE --> E : arrivéeProposition
    DE --> EP : arrivéeProposition
    EP --> DE : refusAffectation
    EP --> E : attributionPoste
    
```

50

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## 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.

51

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML

## Caractérisation d'une transition (2)

**Personne EnPhaseEmbauche**

52

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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-diagrammes 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.

BTD/DLBP/UML

53

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## Concurrence et synchronisation d'états

```

stateDiagram-v2
    state "DemandeurEmploi" {
        EnCoursInscriptionANPE
        ProspectionJournaux
    }
    state "AuTravail" {
        DemanderCongés
        AssurerFonction
    }
    EnCoursInscriptionANPE --> EnPhaseEmbauche : propositionANPE
    ProspectionJournaux --> EnPhaseEmbauche : propositionEntrevue
    DemanderCongés --> EnCongés : acceptationCongés
    AssurerFonction --> EnCongés : terminaisonTâches
    
```

BTD/DLBP/UML

54

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

BTD/DLBP/UML

55

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

(1) + (3)

□ : activité  
 □ (small) : sous processus  
 □ (with dot) : objet  
 → : flux de contrôle  
 - - - - - : flux des artefacts  
 ↓ ↓ : mise en parallèle  
 ↓ ↓ : synchronisation

Opération : identifier client

BTD/DLBP/UML

56

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## 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

57

CENTRALE  
L Y O N


Bertrand DAVID : Développement de logiciels  
basé sur des processus

BTD/DLBP/UML

## UML - Le langage Objet unifié

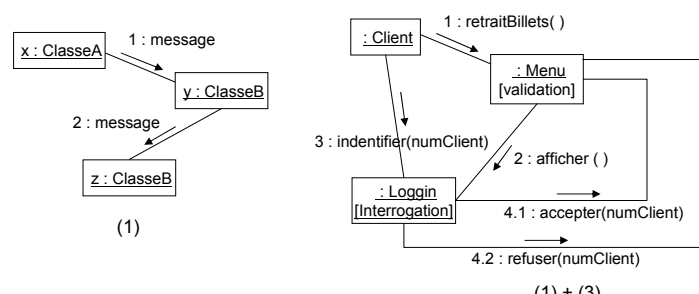
Les diagrammes d'UML

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




58

CENTRALE  
L Y O N

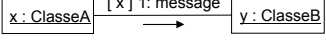
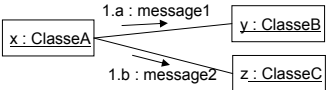
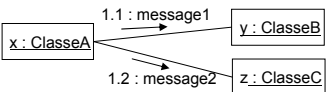
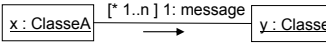
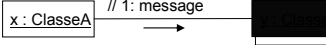
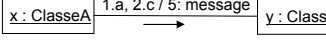
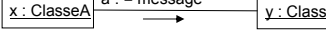
## UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes de collaboration - conventions



Bertrand DAVID : Développement de logiciels basé sur des processus

	message soumis à la condition x
	message1 et message 2 en parallèle
	choix entre message1.1 et message1.2
	message envoyé n fois
	message envoyé en parallèle à plusieurs instances de la classe B
	message envoyé en parallèle à plusieurs instances de la classe B
	a récupère la valeur renvoyée par l'exécution du message

59

CENTRALE  
L Y O N

## Diagramme de collaboration entre objets

Bertrand DAVID : Développement de logiciels basé sur des processus

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.

60

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

**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 »)

BTD/DLBP/UML

61

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

**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

BTD/DLBP/UML

62

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## UML - Le langage Objet unifié

Les diagrammes d'UML  
Diagrammes de composants

technique

Spécif. techniques architecture logicielle

IHM Application Métier Accès données << package >> Persistence

Spécif. fonctionnelles architecture « conceptuelle » « fonctionnelle »

« Fonctionnel » << package >> << catégorie >>

Architecture d'exploitation

Modules

Axe d'intégration

Spécification Implémentation

BTD/DLBP/UML

63

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## 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


BTD/DLBP/UML

64

CENTRALE LYON

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML



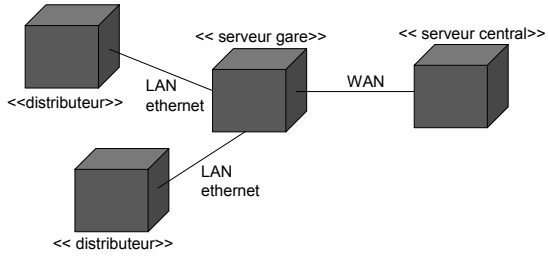
## UML - Le langage Objet unifié

Les diagrammes d'UML

Diagrammes de déploiement

---

**But :**  
Décrire l'architecture matérielle



```

    graph LR
      D1[<< distributeur >>] --- LAN1[LAN ethernet] --- SG[<< serveur gare >>]
      D2[<< distributeur >>] --- LAN2[LAN ethernet] --- SG
      SG --- WAN[WAN] --- SC[<< serveur central >>]
    
```


BTD/DLBP/UML

65

CENTRALE LYON

Bertrand DAVID : Développement de logiciels basé sur des processus

BTD/DLBP/UML



## UML - Le langage Objet unifié

Les diagrammes d'UML

Autres concepts

---

► Stéréotype : mécanisme d'extension d 'UML

<<document électronique>>  
**Facture**

→

@  
**Facture**

Induit certaines propriétés et certains comportement spécifiques

► Interface : « façade » - ensemble de services

<<Interface>>  
**OuvreBouteille**  
 + ôterBouchon ( )

**TireBouchon**  
 + ôterBouchon ( )

**CoûteauSuisse**  
 + ôterBouchon ( )  
 + ouvrirBoîte ( )  
 + piquerAliment ( )  
 + couper ( )

OuvreBouteille

**TireBouchon**  
 + ôterBouchon ( )

BTD/DLBP/UML

66

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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

BTD/DLBP/UML 67

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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.

BTD/DLBP/UML 68

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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.

BTD/DLBP/UML 69

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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.

BTD/DLBP/UML 70

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## 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*.

BTD/DLBP/UML

71

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## 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.

BTD/DLBP/UML

72

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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

BTD/DLBP/UML 73

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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

BTD/DLBP/UML 74

CENTRALE L Y O N	<h2>Les stéréotypes</h2>
Bertrand DAVID : Développement de logiciels basé sur des processus	<ul style="list-style-type: none"><li>● ils classifient des éléments du modèle. Ils permettent donc de créer des familles d'éléments associés à un même stéréotype ;</li><li>● ils modifient la sémantique des éléments associés et permettent la création de nouveaux concepts propres à une application.</li></ul> <p>Attention toutefois à l'intégrité du métamodèle de UML.</p>
BTD/DLBP/UML	75

CENTRALE L Y O N	<h2>Les contraintes</h2>
Bertrand DAVID : Développement de logiciels basé sur des processus	<p>La contrainte permet de préciser le contexte du modèle en positionnant des restrictions.</p>
BTD/DLBP/UML	76

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## 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.


BTD/DLBP/UML 77

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels basé sur des processus

## 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

BTD/DLBP/UML 78

CENTRALE  
L Y O N

## 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 spéc. UML (1.4)
Modélisation de collecticiels	Diag. De collaboration, de classes

BTD/DLBP/UML

79

CENTRALE  
L Y O N

## Approche Objet et formalismes UML

**Plan**

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

BTD/DLBP/UML

80

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## 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
- ...

BTD/DLBP/UML 81

CENTRALE  
L Y O N

Bertrand DAVID : Développement de logiciels  
basé sur des processus

## AGL UML

### Quelques AGL :

- Together ( Togethersoft) <http://www.togethersoft.com>
- Rational ROSE (Rational corp.) <http://www.rational.com>
- Paradigm (Platinum) <http://www.platinum.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>

BTD/DLBP/UML 82