



03

Chapitre

Un formalisme : UML

112AC1 : Génie logiciel et Conception orientée objet

Régis Clouard, ENSICAEN - GREYC

« Si les ouvriers construisaient les bâtiments
comme les développeurs écrivent leurs programmes,
le premier pic-vert venu aurait détruit toute civilisation. »

Gerald Weinberg

Plan du chapitre

2

1

Modélisation
avec UML

Un formalisme de modélisation

3

- Intention : Décrire le logiciel tout au long de son cycle de vie avec un haut niveau d'abstraction
- **UML (Unified Modeling Language)**
 - Visuel (à base de diagrammes)
 - Indépendant de tout langage de programmation
 - Correspondance forte avec les langages de programmation
 - ▶ UML ↔ Java, C++, Php, Python...



Grady Booch



James Rumbaugh



Ivar Jacobson

Plan du chapitre

1

Modélisation
avec UML

2

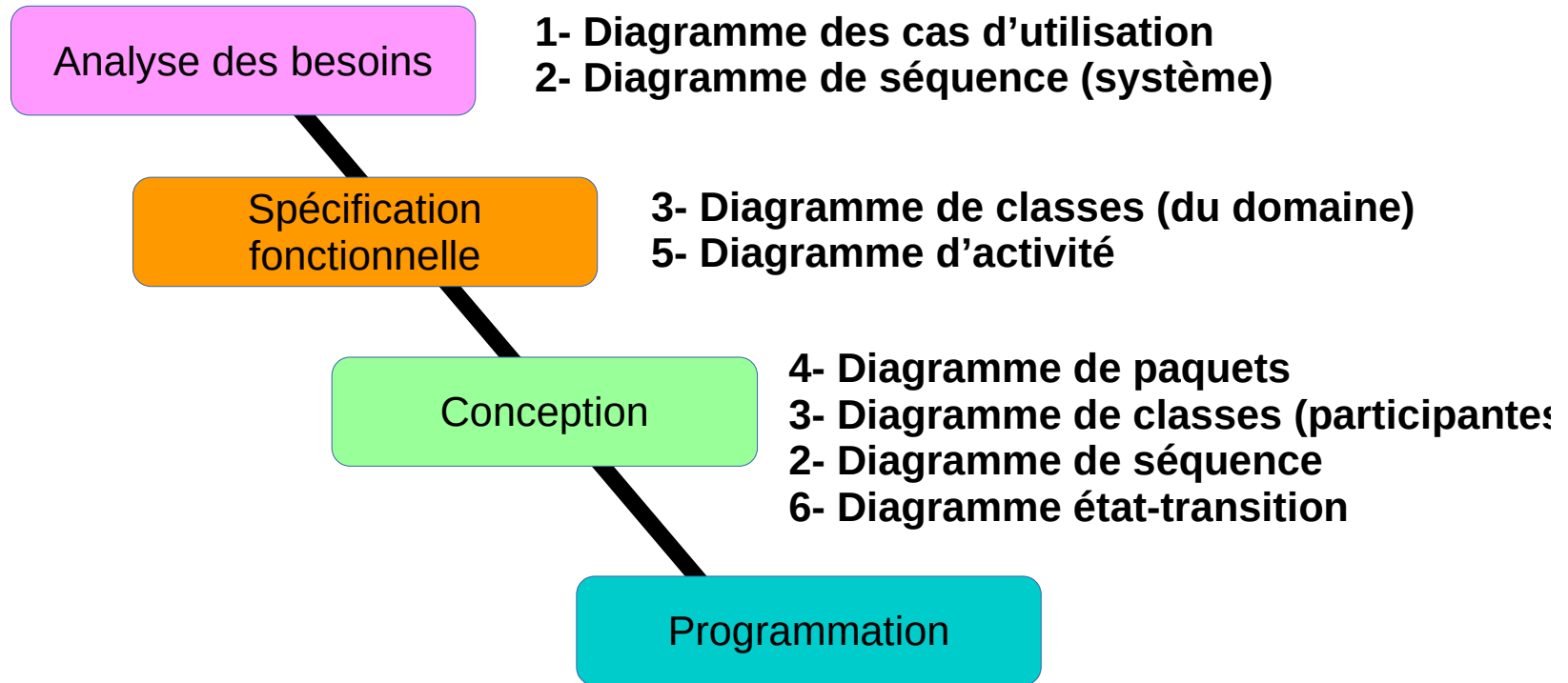
Les principaux
diagrammes

Principaux diagrammes

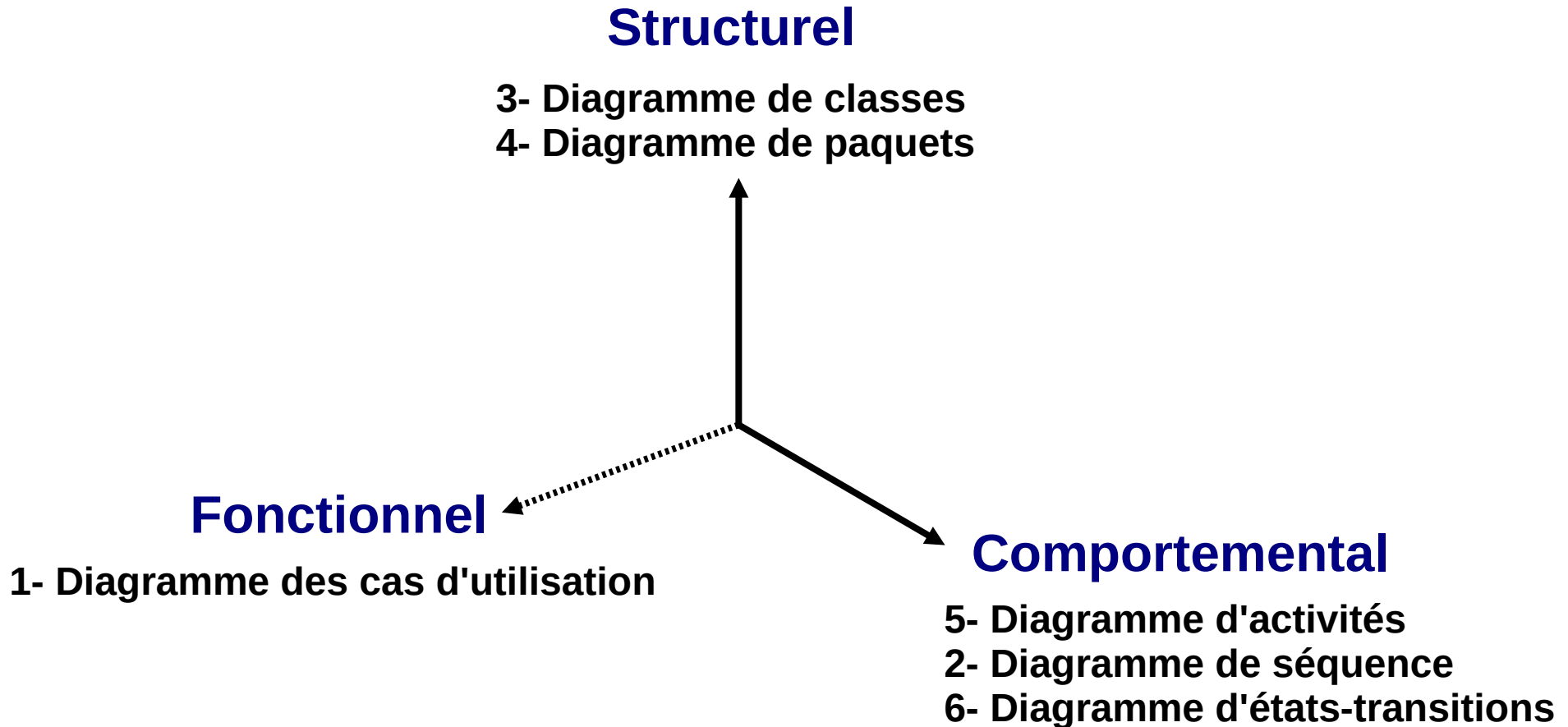
- 6 diagrammes / 13 normalisés
 - 1) Diagramme des cas d'utilisation
 - 2) Diagramme de séquence
 - 3) Diagramme de classe
 - 4) Diagramme de paquet
 - 5) Diagramme d'activité
 - 6) Diagramme état-transition

Modélisation avec UML

- Cycle en cascade (cycle obsolète mais pédagogique)



Trois vues sur la modélisation



Exemple fil rouge

- Système simplifié de **GAB** qui offre les services suivants :
 - 1) Le distributeur délivre de l'argent à tout porteur d'une carte VISA.
 - 2) Pour les clients porteurs d'une carte de crédit de la banque, il est possible de consulter le solde de son compte, de déposer du numéraire ou des chèques.
 - 3) Toutes les transactions sont sécurisées.
 - 4) Dans le cas où une carte est avalée, un opérateur de maintenance se charge de la récupérer. C'est la même personne qui collecte les dépôts d'argent et qui recharge le distributeur.

1/ Diagramme des cas d'utilisation

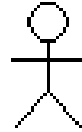
- Intention
 - Définir les fonctionnalités du futur système.
 - ▶ Qu'est ce que l'utilisateur peut faire avec le logiciel ?
- Décrire quoi faire (pas comment le faire)
- En dépit de son caractère apparemment trivial, il est fondamental pour toutes les phases et tous les niveaux de la modélisation

Diagramme des cas d'utilisation : syntaxe

10

■ Acteur

- Humain ou un système tier que l'on développe pas



■ Cas

- Fonctionnalité du système.



■ Relations

- Include (**obligatoire**)
- Extends (**facultatif**)
- héritage

`<<include>>`
----->

`<<extend>>`
----->



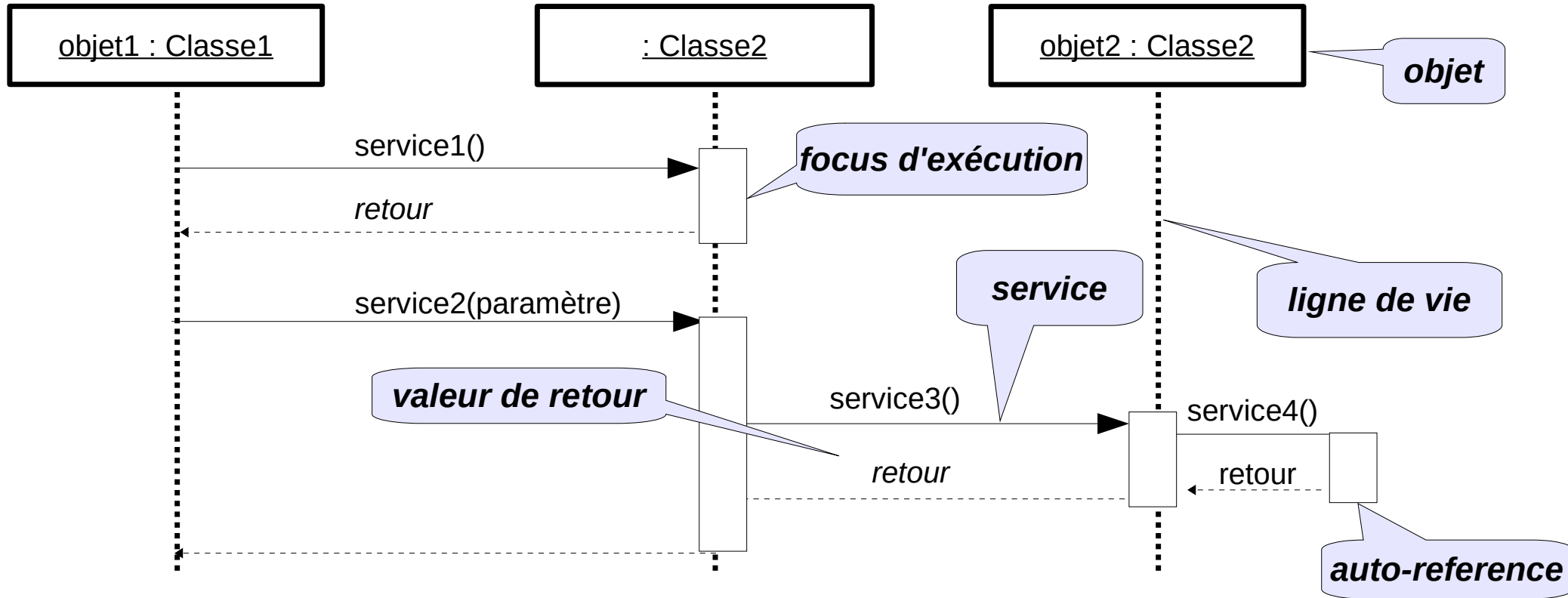
2/ Diagramme de séquence

■ Intention

- Le diagramme des cas d'utilisation ne donne pas l'ordre entre les cas
- C'est le rôle du diagramme de séquence
 - ▶ Décrit la séquence d'échange des services entre les acteurs et le système

Diagramme de séquence : syntaxe

12



3/ Diagramme de classe (domaine)

13

- Intention
 - Difficile de trouver les classes a priori
 - ▶ S'appuyer sur celles du **domaine / métier**
 - ▶ Enrichir ensuite avec les classes liées aux choix d'implémentation

Diagramme de classe : syntaxe



Relations :



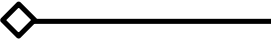


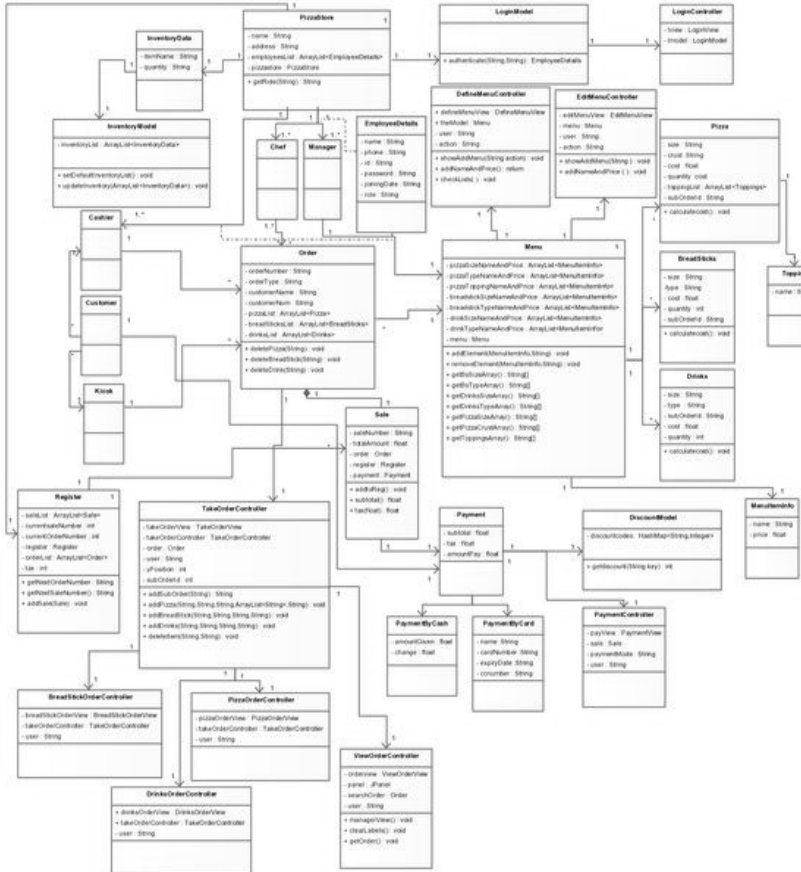
- Héritage 
- Association 
- Agrégation 
- Composition 
- Dépendance 
- Réalisation 

Diagramme de classes

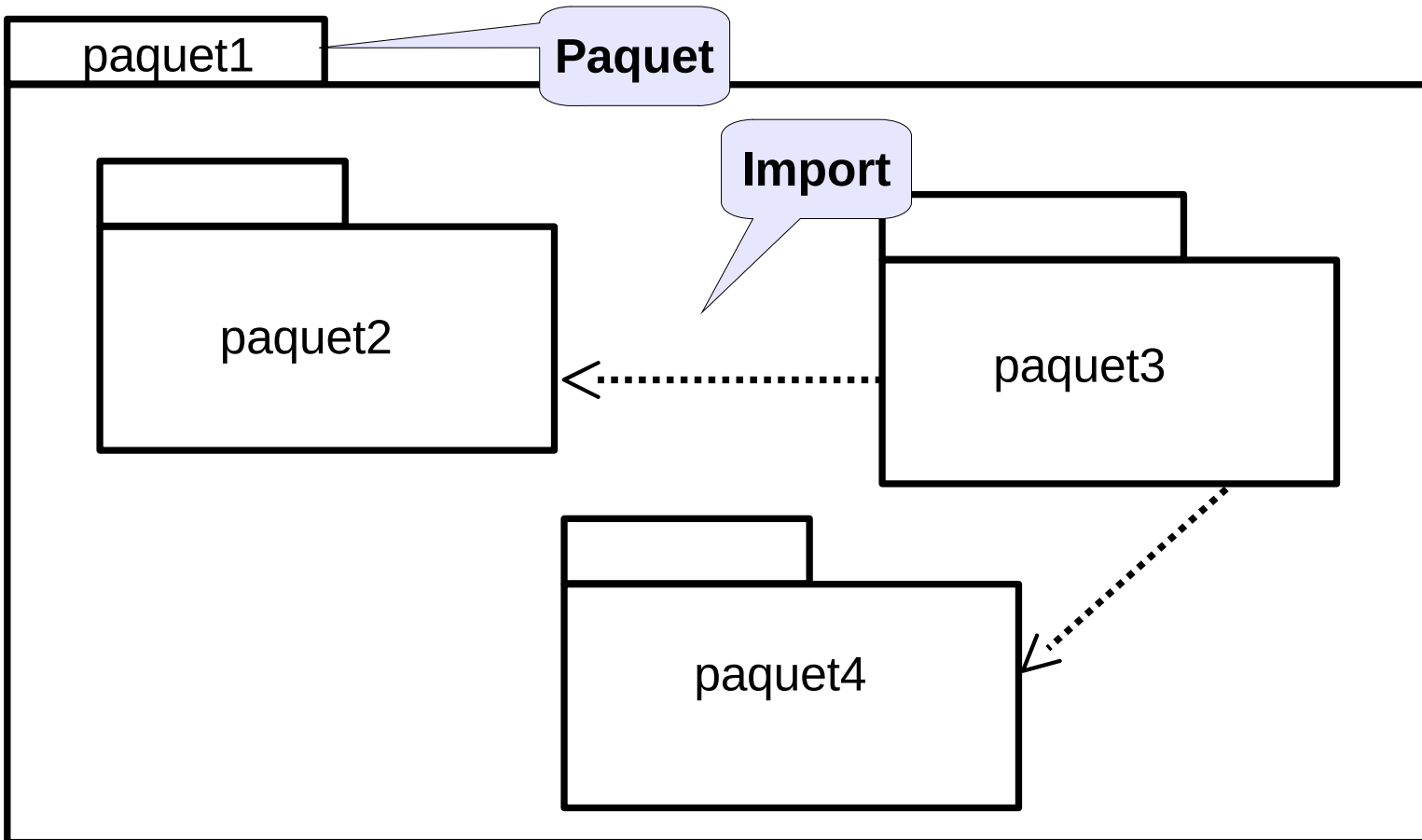
- Ne mettez **jamais** les attributs (ni les méthodes)
- Attention à la lisibilité !
 - Découper en plusieurs diagrammes



4/ Diagramme de paquet

- Intention
 - Vision organisationnelle du projet en dossiers et sous-dossiers
 - ▶ Reflet de l'architecture ou la carte de construction du logiciel
- Paquet = dossier avec plusieurs classes

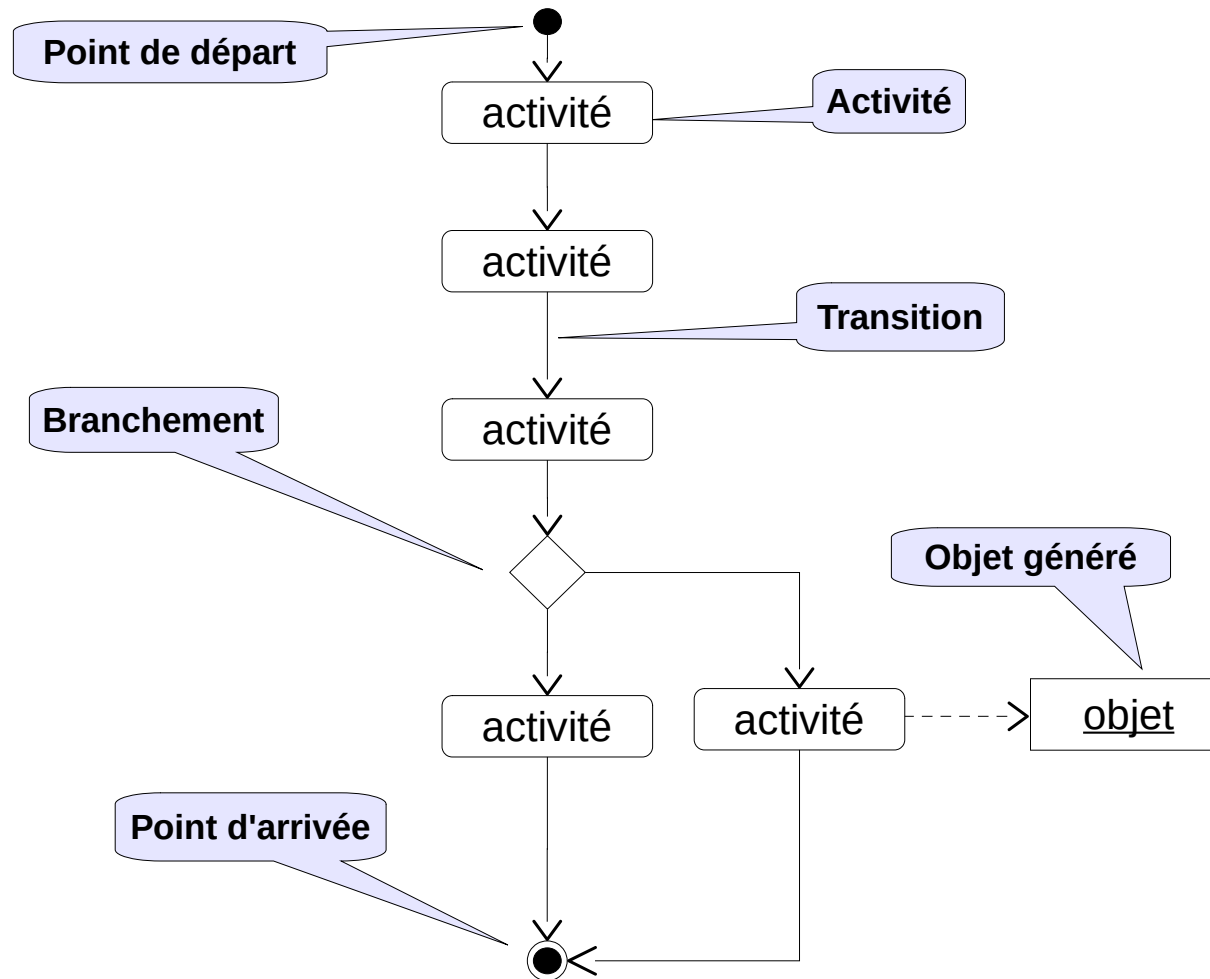
Diagramme de paquets : syntaxe



5/ Diagramme d'activité

- Intention
 - Modéliser un traitement
 - ▶ Un cas d'utilisation (gros grain)
 - ▶ Un algorithme pour une méthode (grain fin)
- Il est indépendant des classes
 - Peut être réalisé en parallèle des diagrammes impliquant les classes

Diagramme d'activités : syntaxe



2/ Diagramme de séquence

20

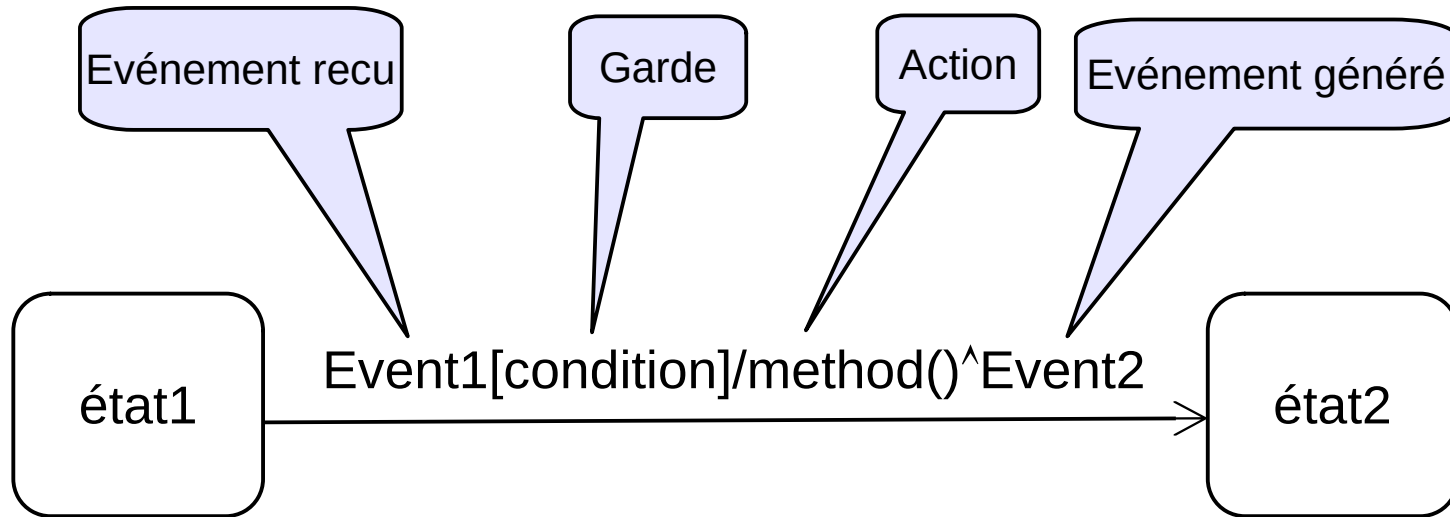
- Intention
 - Expliciter les échanges de services entre les classes
 - Niveau de granularité très fin

6/ Diagramme état-transition

21

- Décrire
 - les différents états que peut prendre **une classe** un peu complexe
 - Les actions qui provoquent un changement d'état

Diagramme d'états-transitions : syntaxe



Plan du chapitre

1
Modélisation
avec UML

2
Les principaux
diagrammes

3
Conclusion

Atelier de génie logiciel (AGL - *case tools*)

24

- DOUML
- ArgoUML
- StartUML
- <https://www.draw.io/>

Que retenir de ce chapitre ?

- UML est un langage de modélisation diagrammatique :
 - Très puissant.
 - Couvre tout le cycle de vie.
 - Incontournable pour la modélisation de logiciels.
 - International et normalisé.
- Il n'y a que 6 diagrammes principaux
 - Cas d'utilisation
 - Classe
 - Activité
 - Séquences
 - Paquet
 - État-transition
- Les autres diagrammes sont plus spécialisés (temps réel, déploiement, infrastructure) ou des variantes de ceux présentés (communication, interaction)