ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

03
Chapitre

Un formalisme : UML

1I2AC1 : Génie logiciel et Conception orientée objet

Régis Clouard, ENSICAEN - GREYC

« Il existe deux manières de concevoir un logiciel. La première, c'est de le faire si simple qu'il est évident qu'il ne présente aucun problème. La seconde, c'est de le faire si compliqué qu'il ne présente aucun problème évident. La première méthode est de loin la plus complexe. »

Antony R. Hoare, prix Turing 1980

Plan du chapitre

1

Modélisation avec UML

Un formalisme de modélisation

- 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

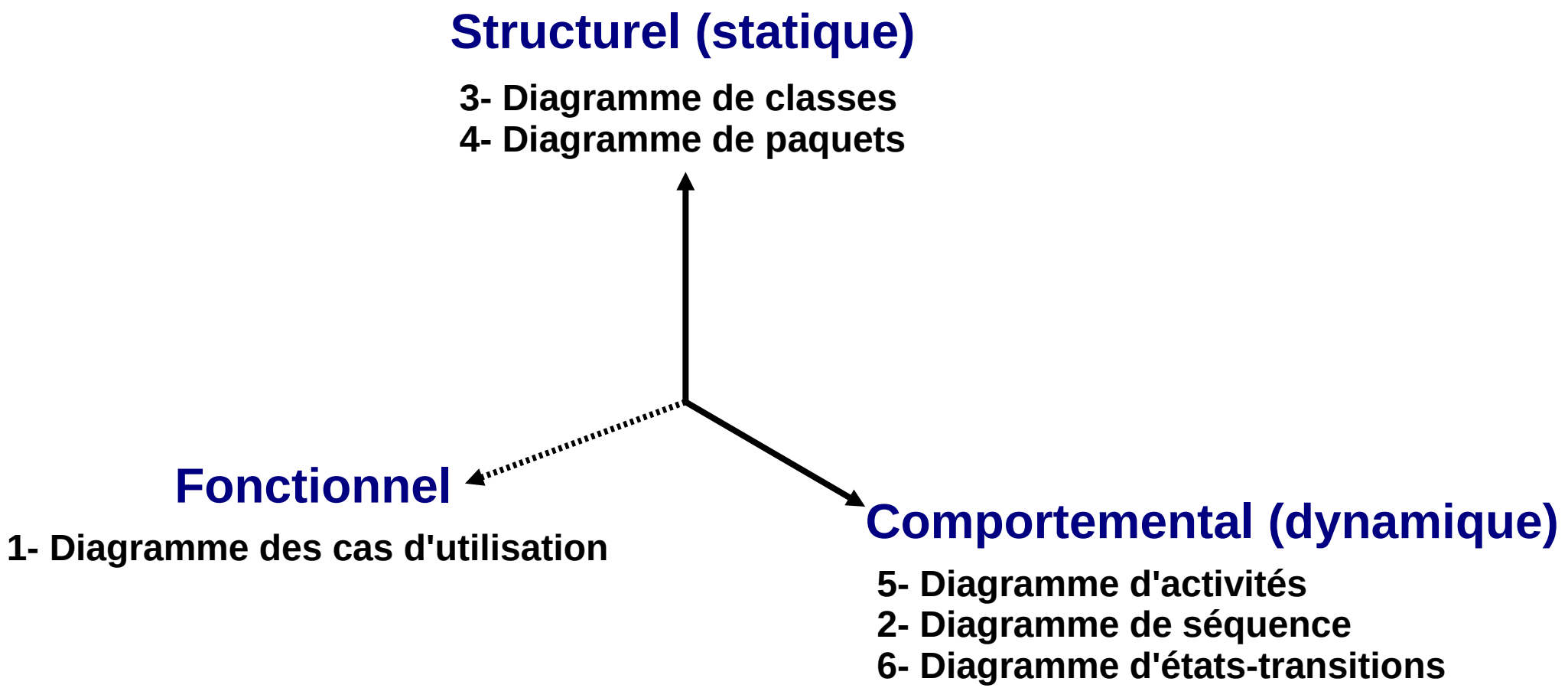
Principaux diagrammes

- UML définit 13 normalisés
- Ce cours ne considère que les 6 diagrammes principaux :
  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
- Les autres diagrammes sont soit plus spécialisés (temps réel, déploiement, infrastructure) soit des variantes de ceux présentés (communication, collaboration, interaction)

6 diagrammes pour tout modéliser

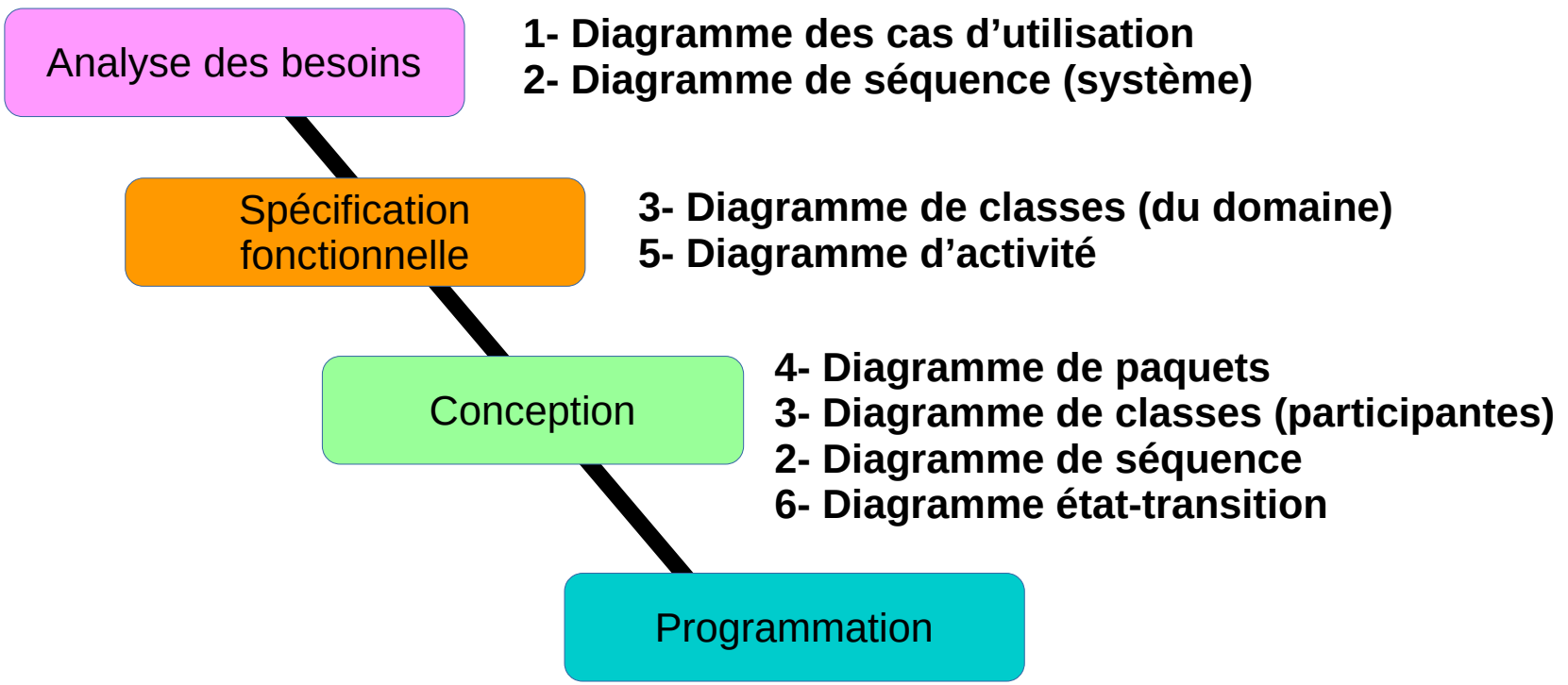
- Les diagrammes permettent de comprendre le problème
  - Élaborés avec le client
- Les diagrammes permettent de décrire la solution proposée
  - Description de la solution à plusieurs niveaux d'abstraction : architecture et technique

Trois vues sur la modélisation



Modélisation avec UML

- Cycle en cascade (cycle obsolète mais pédagogique)



Exemple fil rouge

- Système simplifié de **GAB** (Guichet Automatique de Banque) qui offre les services suivants :

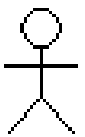
1) Le distributeur **délivre** de l'argent à tout porteur d'une **carte VISA**.
2) Pour les clients porteurs d'une carte de crédit de la banque, il est possible de **consulter** le solde de son compte, de **déposer** du numéraire ou des chèques.
3) Toutes les transactions sont sécurisées que ce soit avec le **service d'authentification (SA) Visa**, pour les transactions de retraits effectuées avec une carte Visa ou avec le **système d'information (SI)** de la banque pour autoriser toutes les opérations effectuées par un client avec sa carte de banque.
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

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



- Cas
  - Fonctionnalité du système.

- Relations
  - Include (**obligatoire**)

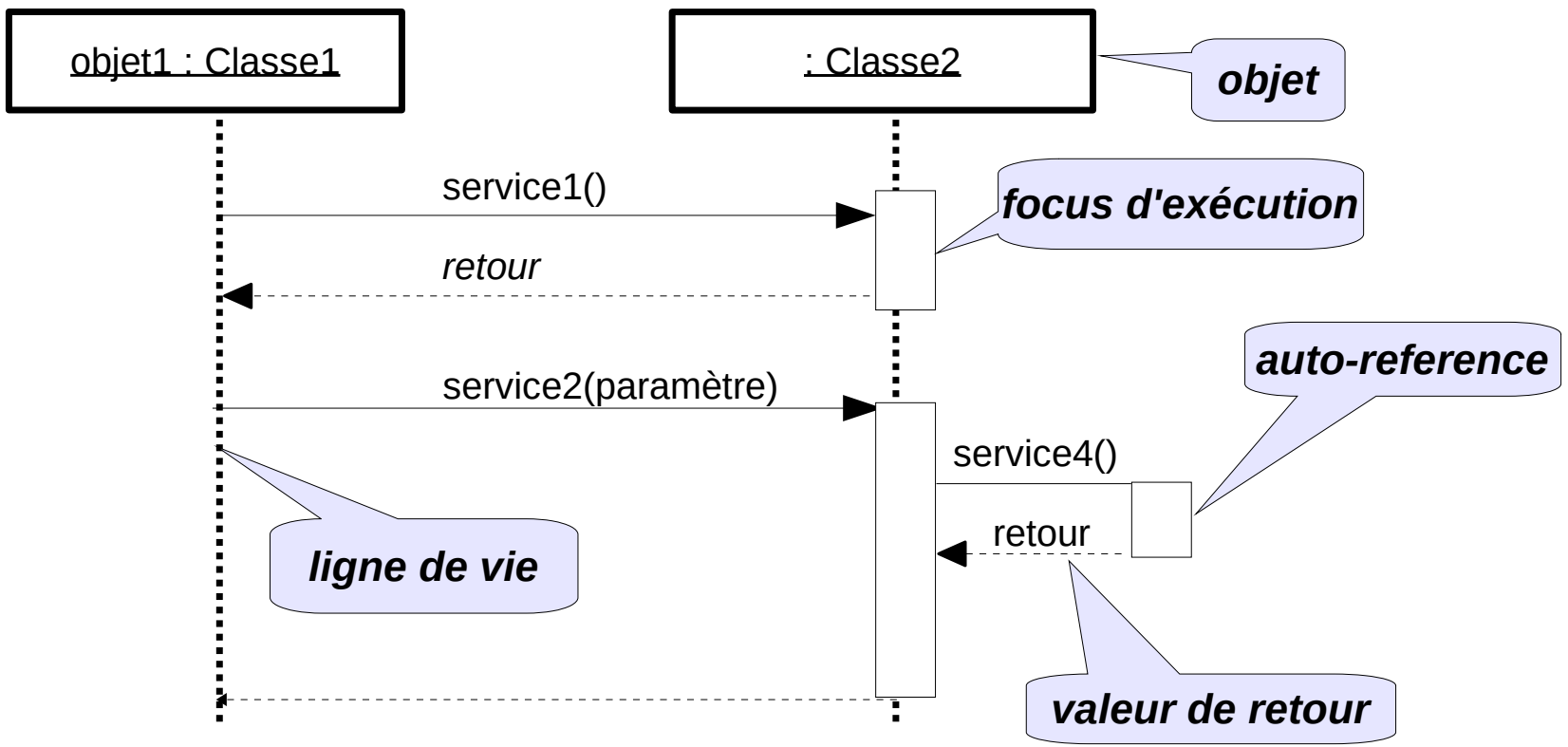
  - Extends (**facultatif**)

  - héritage

2/ Diagramme de séquence

- Diagramme de séquence système
- 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



3/ Diagramme de classe (domaine)

- 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

Classe

Relations :

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

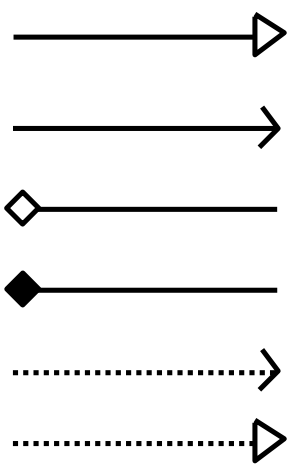
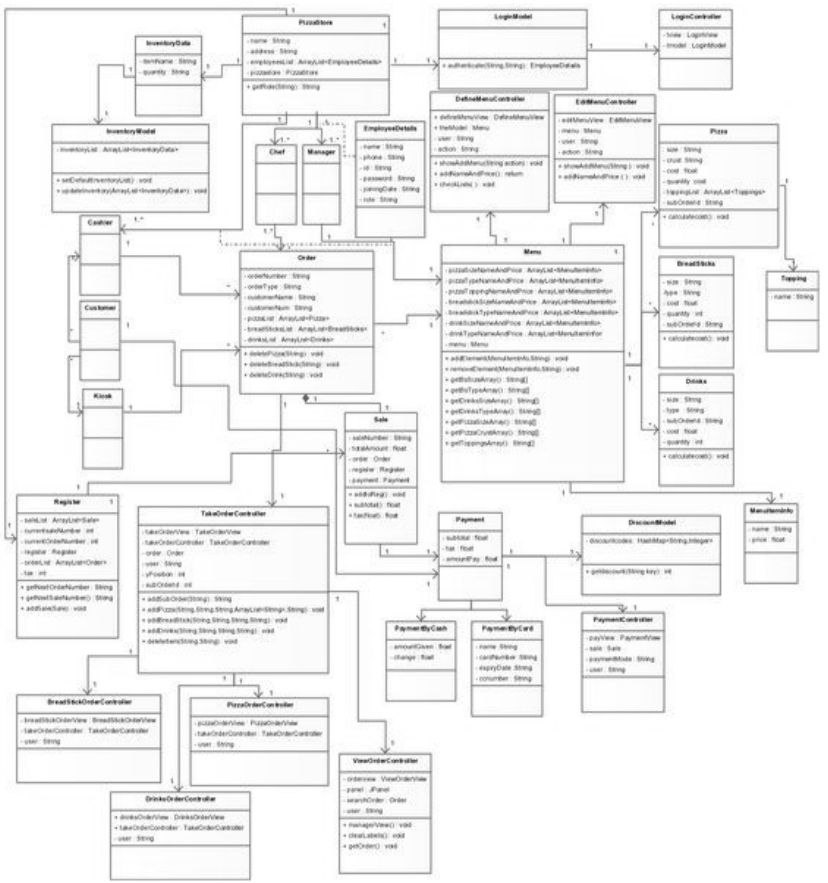


Diagramme de classes

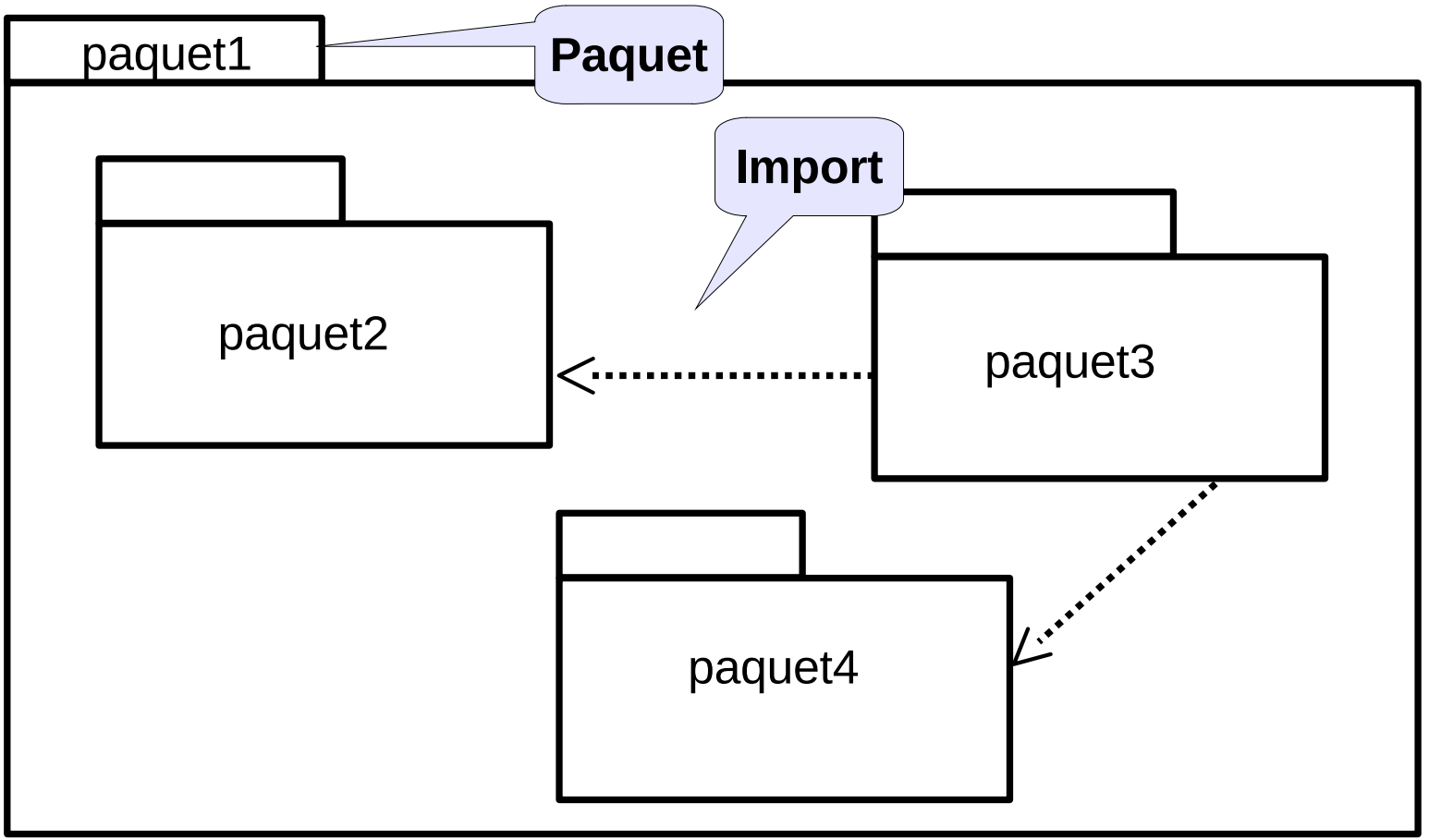
- Ne mettez **jamais** les attributs (ni les méthodes)
- Attention à la lisibilité !
  - Découper en plusieurs diagrammes



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

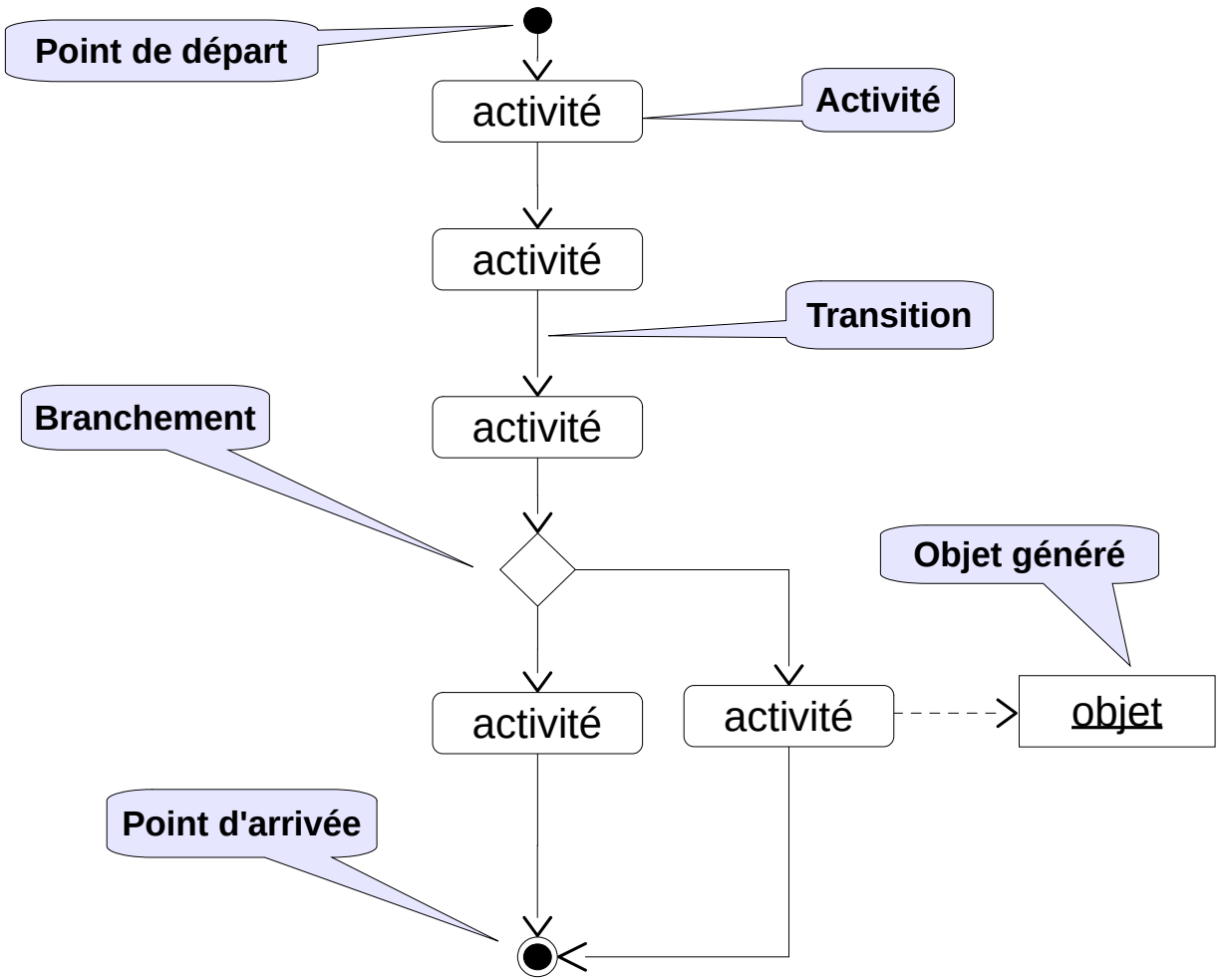
Diagramme de paquets : syntaxe



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

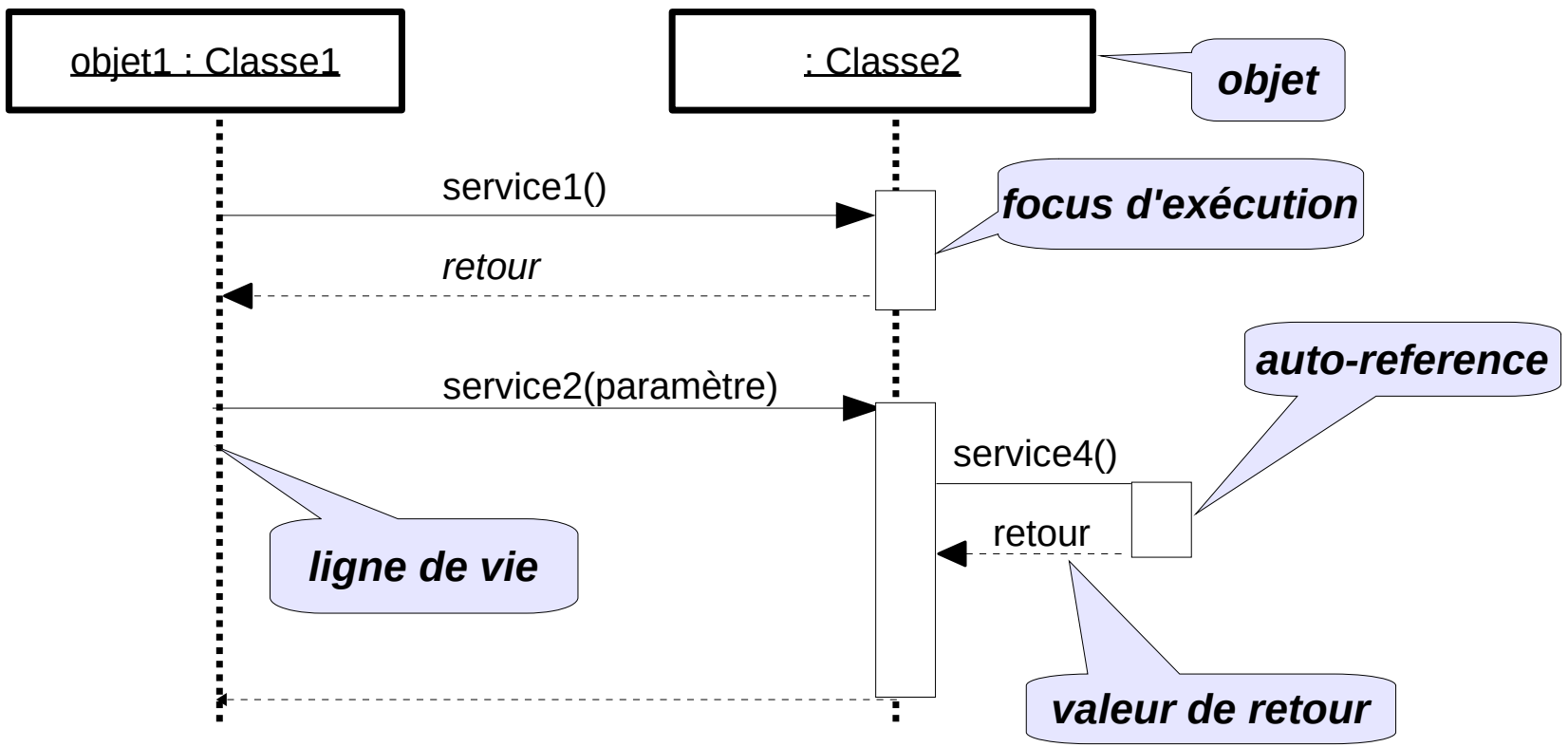
Diagramme d'activités : syntaxe



2/ Diagramme de séquence

- Intention
  - Expliciter les échanges de services entre les classes
  - Niveau de granularité très fin
- Remarque : variante du diagramme de collaboration vu en TP (tondeur de pelouse)

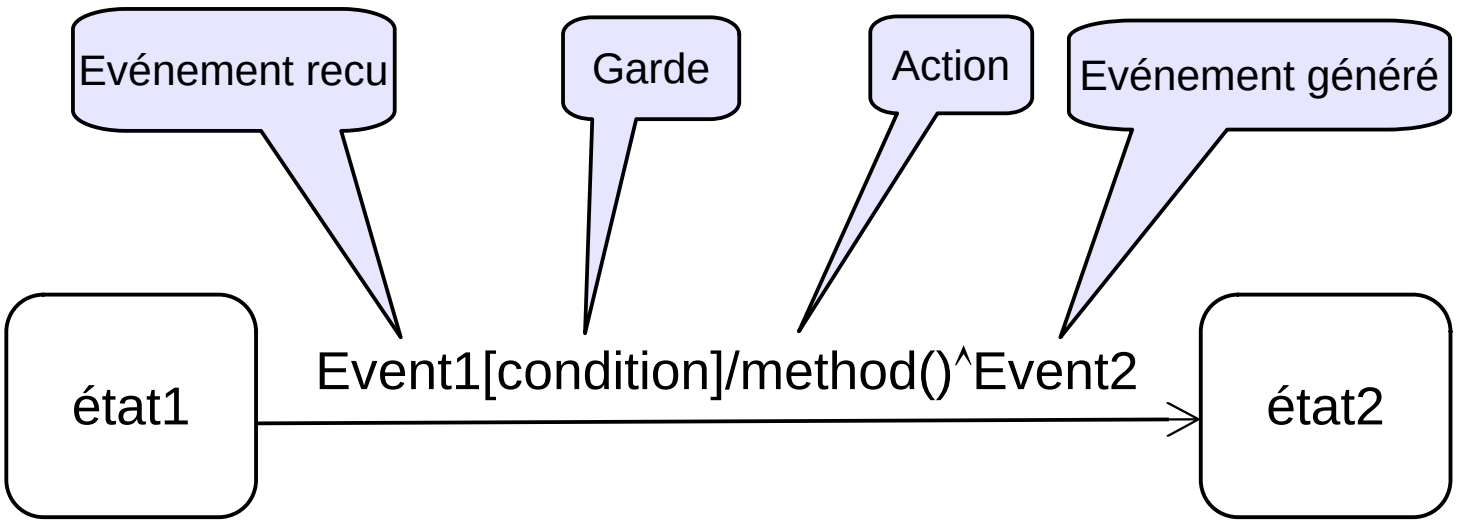
Diagramme de séquence : syntaxe



6/ Diagramme état-transition

- Décrire
  - lLes 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*)

- 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