



03

Chapitre

Un formalisme : UML

112AC1 : 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
Modéliser
avec UML

Un formalisme de modélisation

3

- Intention : Représenter l'analyse et la conception d'un 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
Modéliser
avec UML

2
Les principaux
diagrammes

Principaux diagrammes

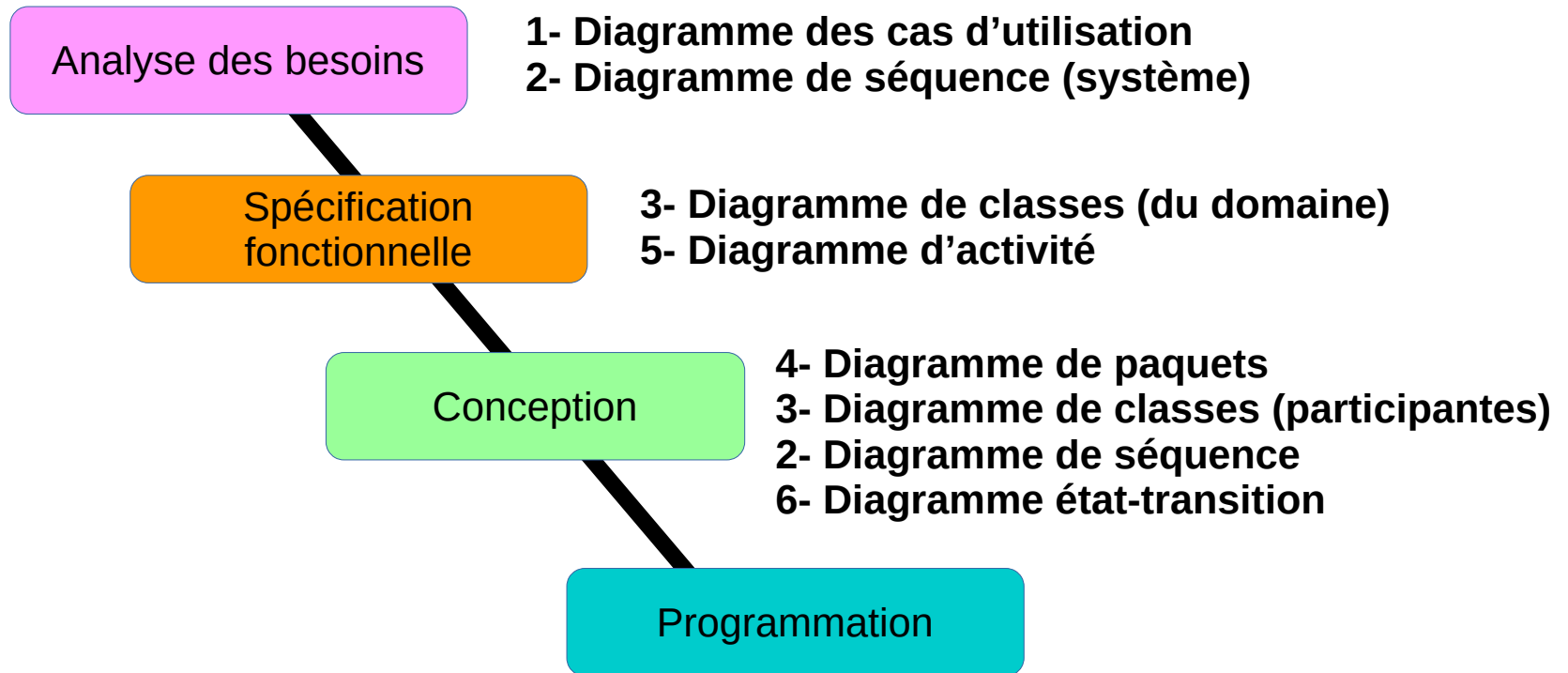
- UML définit 13 normalisés
- Ce cours ne considère que les 6 diagrammes principaux :
 - 1) Diagramme des cas d'utilisation
 - 2) Diagramme de séquence
 - 3) Diagramme de classe
 - 4) Diagramme de paquet
 - 5) Diagramme d'activité
 - 6) Diagramme état-transition
- Les autres diagrammes sont soit spécialisés (temps réel, déploiement, infrastructure) soit des variantes de ceux présentés (communication, collaboration, interaction)

6 diagrammes pour tout modéliser

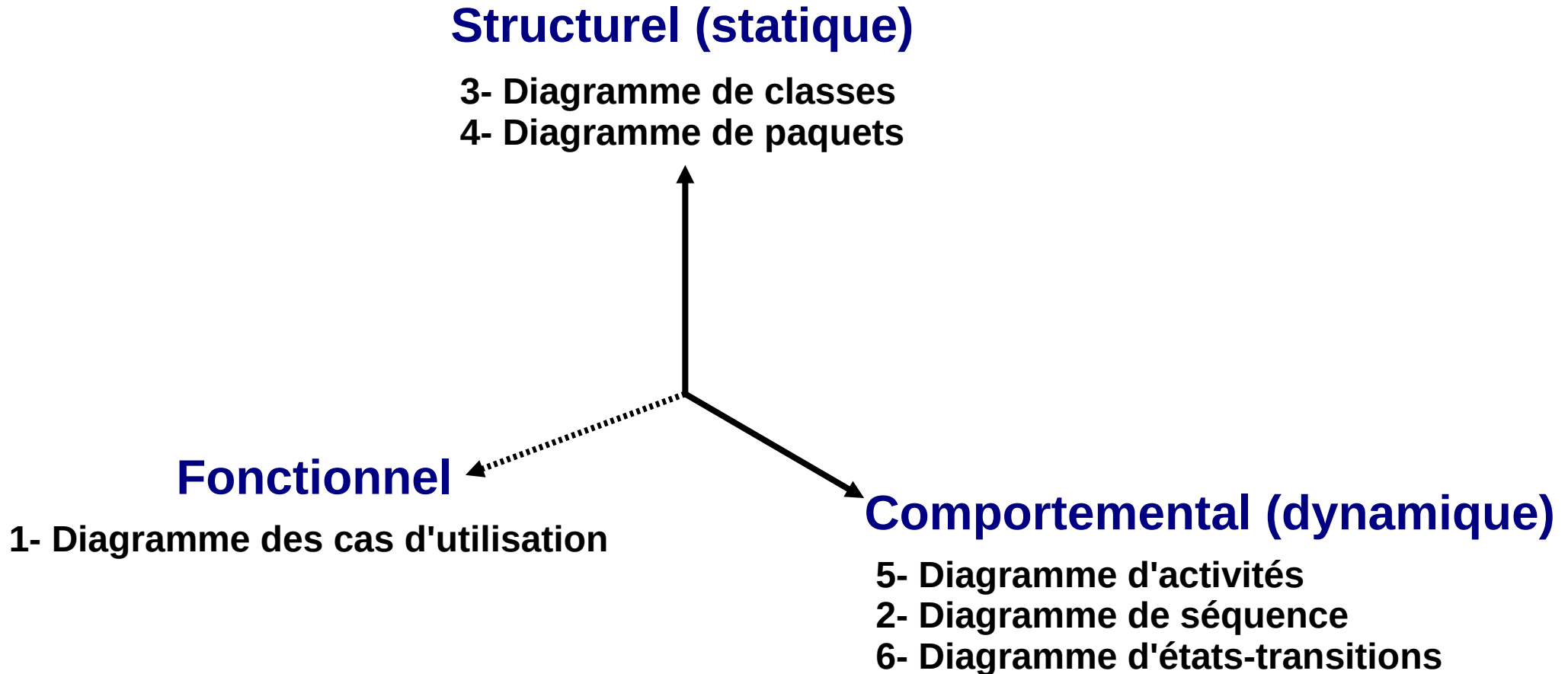
- Les diagrammes permettent de :
 - 1) comprendre le problème (analyse)
 - ▶ Ils sont élaborés avec le client
 - 2) élaborer une solution (conception)
 - ▶ C'est une description de la solution à plusieurs niveaux d'abstraction : architecture, technique et implémentation

Modélisation avec UML

- Cycle en cascade (*cycle obsolète mais pédagogique*)



Trois vues sur la modélisation



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.
 - 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 de la banque (SI)** pour autoriser 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 **recharge** le distributeur en billets.

1/ Diagramme des cas d'utilisation

10

- Intention (point de vue fonctionnel)
 - Définir les fonctionnalités du futur système à travers les **points de contact** entre l'utilisateur et le système
- 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

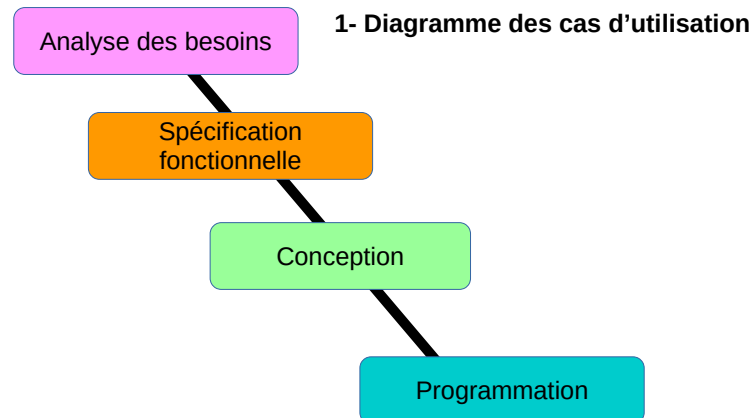
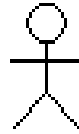


Diagramme des cas d'utilisation : syntaxe

11

■ Acteur

- Humain ou un système tier



■ Cas : point de contact

- un verbe



■ Relations

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

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

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



Demo : GAB

2/ Diagramme de séquence

- Intention (point de vue comportemental)
 - 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

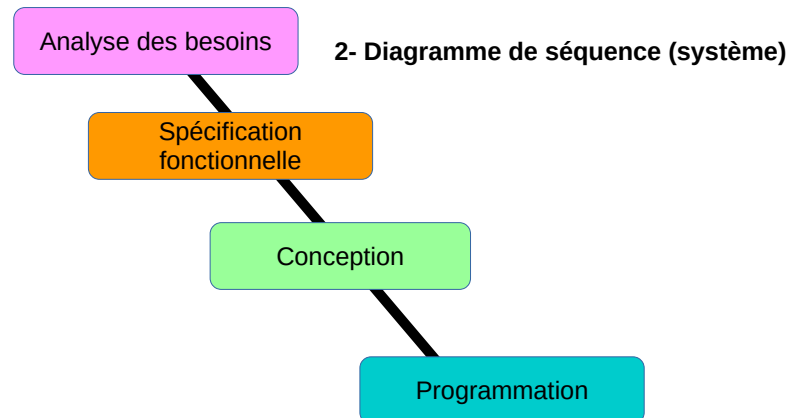
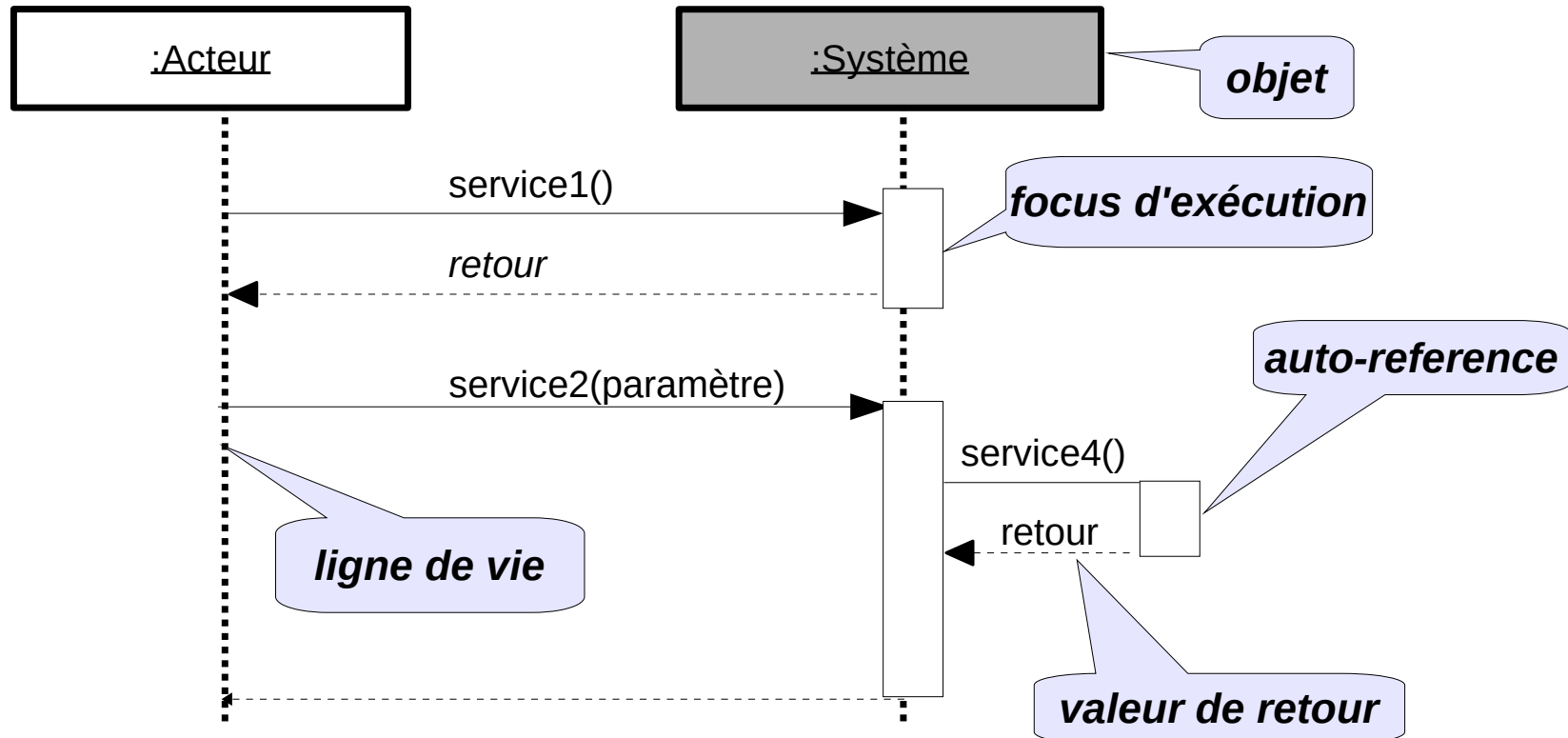


Diagramme de séquence : syntaxe

13



Demo : GAB

3/ Diagramme de classe (domaine)

- Intention (point de vue structurel)
 - Difficile de trouver les classes a priori, pour cela :
 - ▶ s'appuyer sur celles du **domaine** (ie, **classes métier**)
 - ▶ enrichir ensuite avec les classes liées aux choix d'implémentation

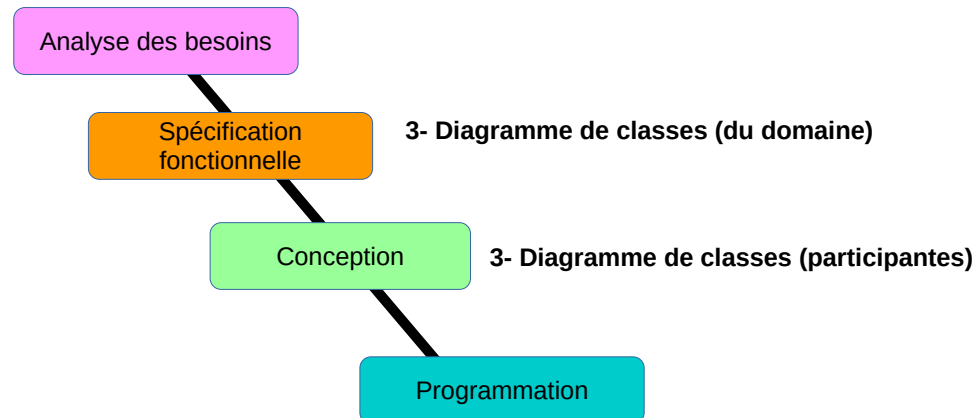





Diagramme de classe : syntaxe

Classe

Relations :

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

4/ Diagramme de paquet

- Intention (point de vue structurel)
 - 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

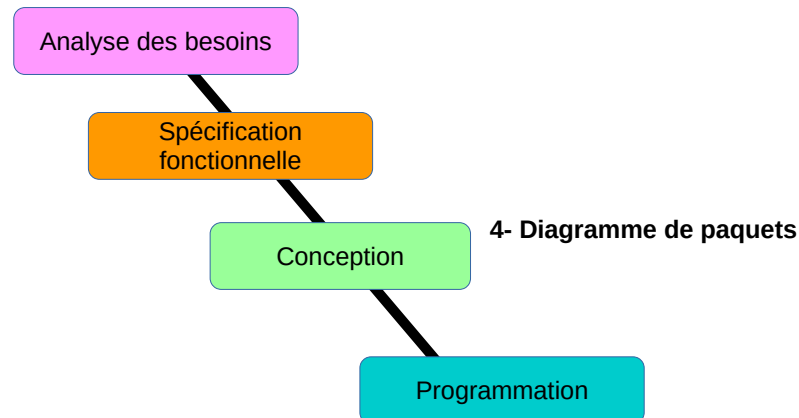
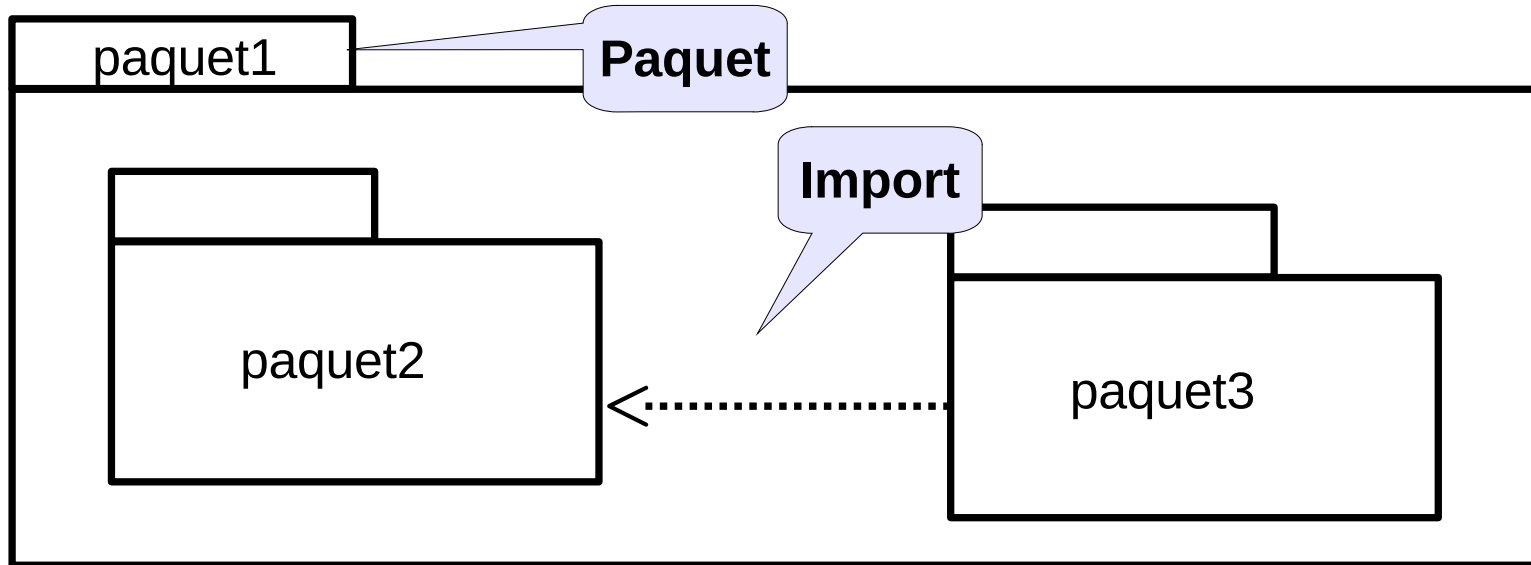


Diagramme de paquets : syntaxe

18



Demo : GAB

5/ Diagramme d'activité

- Intention (point de vue comportemental)
 - Modéliser un traitement, par exemple :
 - ▶ 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

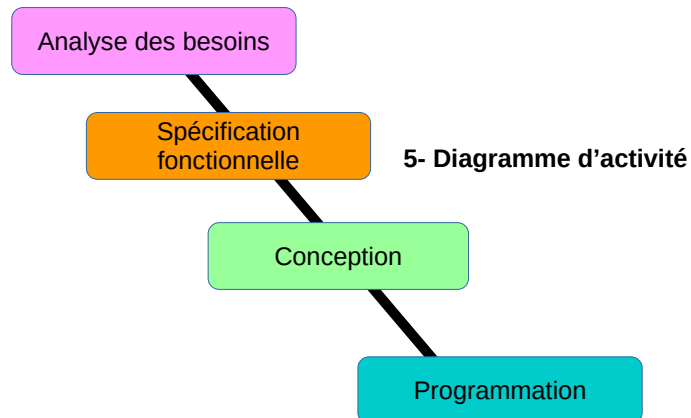
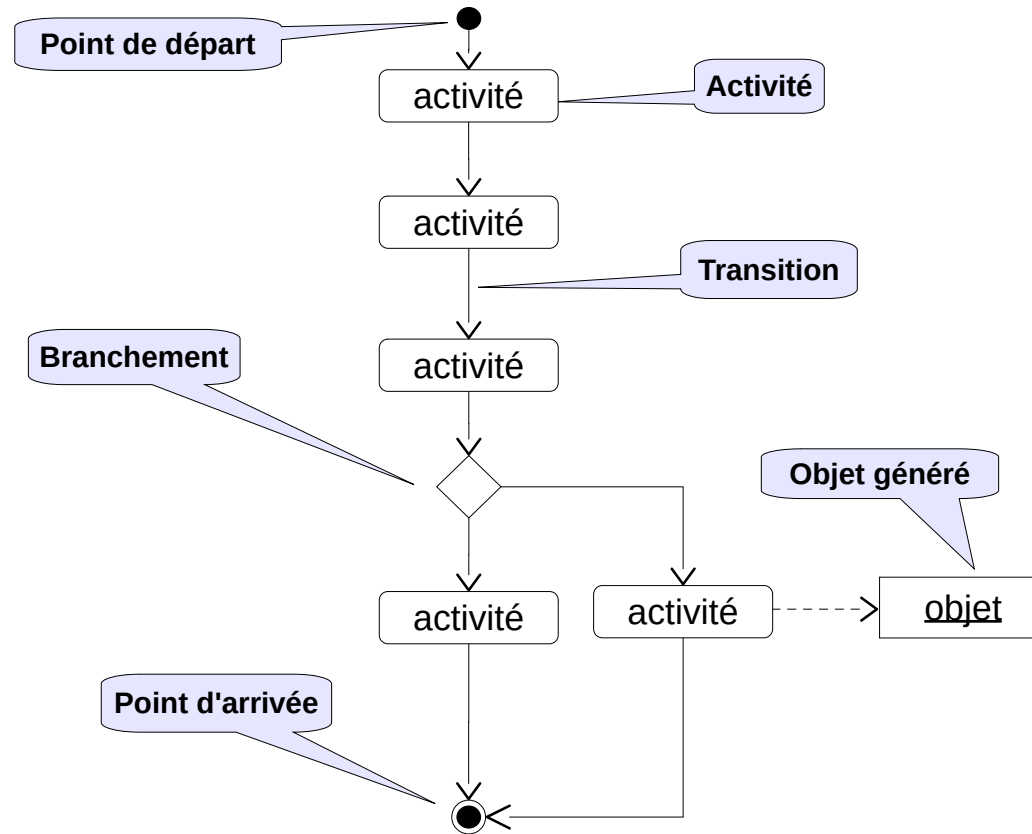


Diagramme d'activités : syntaxe

20



Demo : GAB

2 bis/ Diagramme de séquence

21

- Intention (point de vue comportemental)
 - 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)

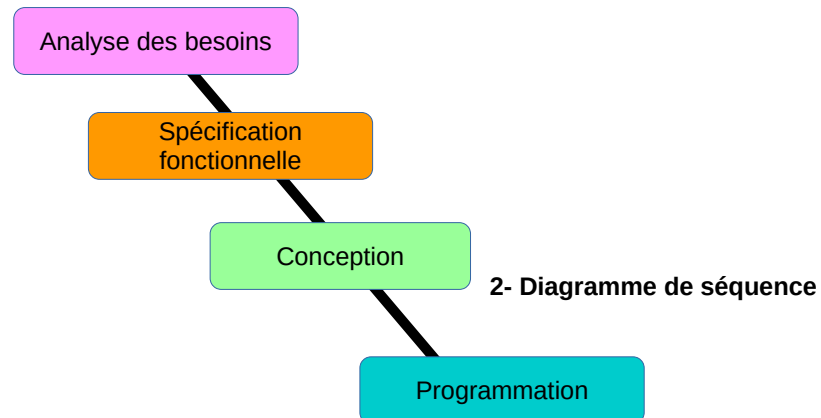
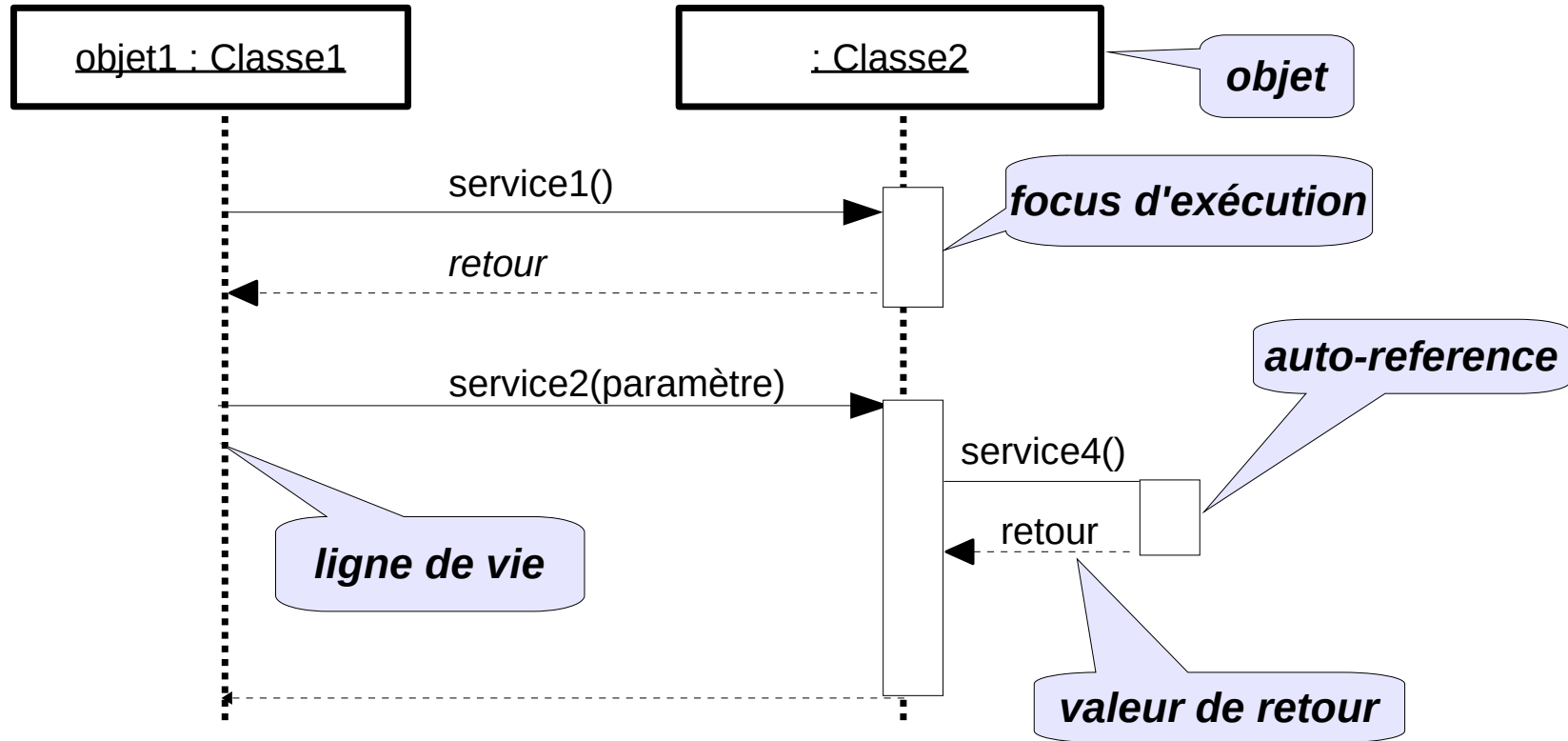


Diagramme de séquence : syntaxe

22



Demo : GAB

6/ Diagramme état-transition

- Intention (point de vue comportemental)
- Décrire
 - Les différents états que peut prendre **une classe** un peu complexe
 - Les actions qui provoquent un changement d'état

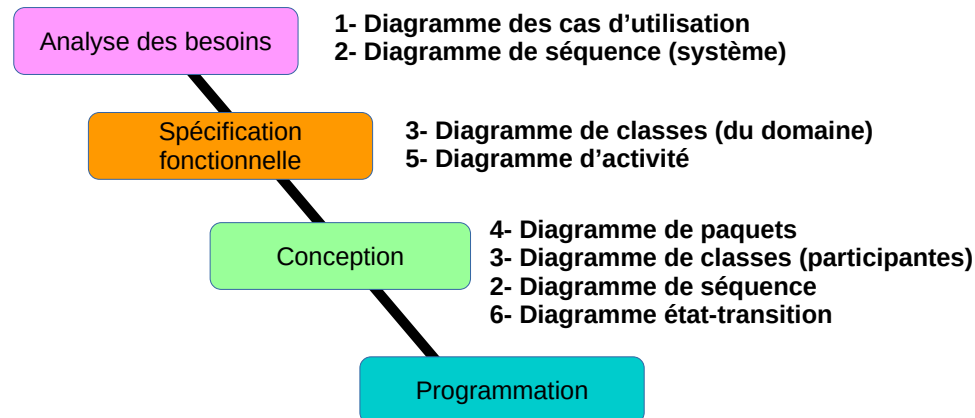
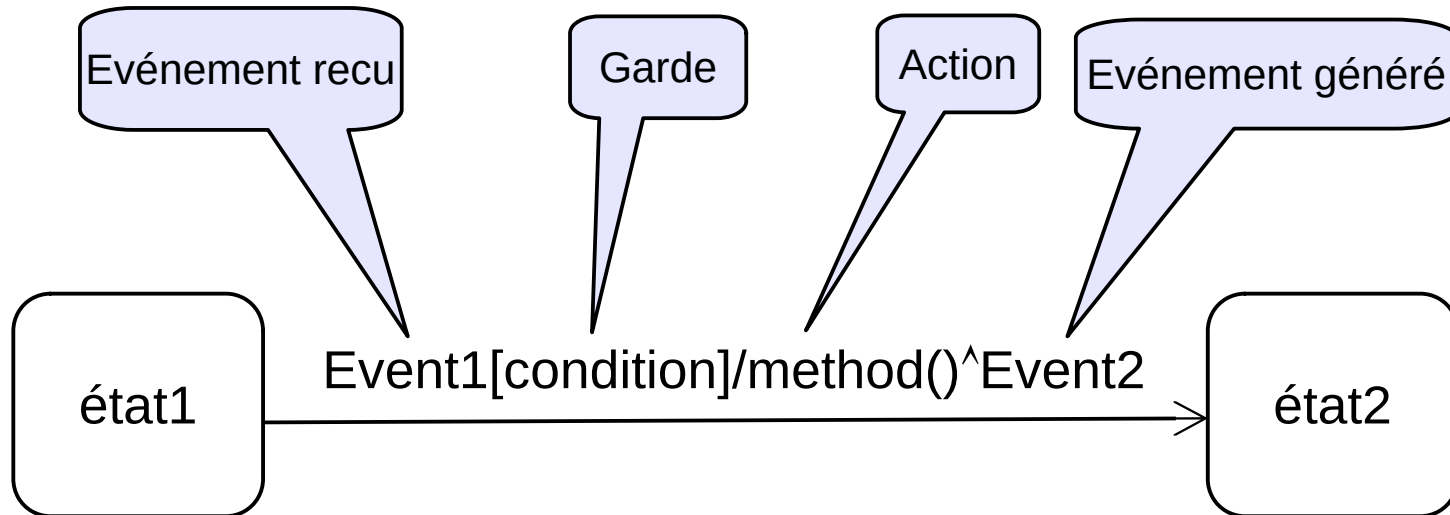


Diagramme d'états-transitions : syntaxe

24



Demo : GAB

Plan du chapitre

1
Modéliser
avec UML

2
Les principaux
diagrammes

3
Conclusion

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

26

- 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