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

« 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

1

Modélisation avec UML

Un formalisme de modélisation

- UML (Unified Modeling Language)
  - **Visuel** (à base de diagrammes)
    - débarrassé des contraintes syntaxiques
  - **Indépendant** de tout langage de programmation
  - **Correspondance** forte avec les langages de programmation
    - UML ↔ Java, C++, Php, Python…

Grady Booch
BOOCH

James Rumbaugh
OMT

Ivar Jacobson
OOSE

Plan du chapitre

1
Modélisation avec UML

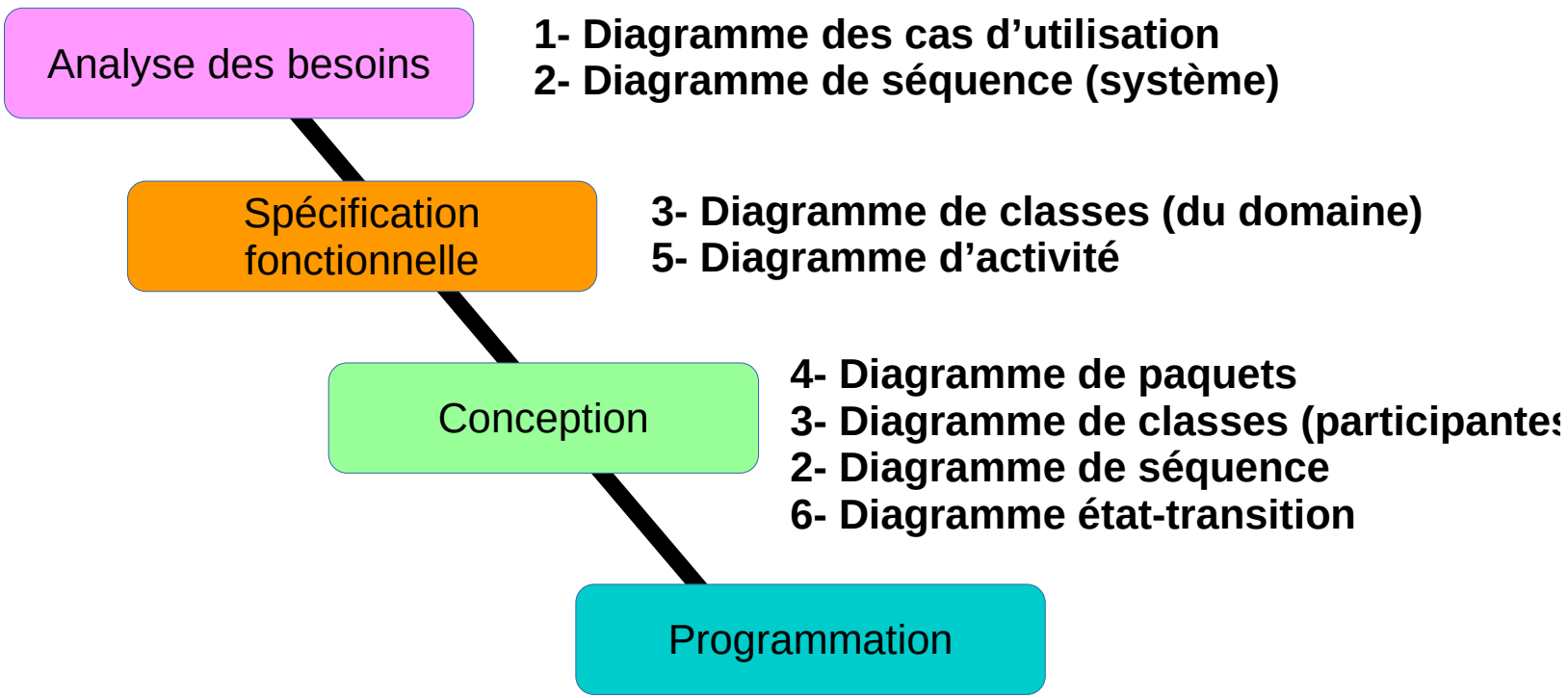
2
Les principaux diagrammes

Principaux diagrammes

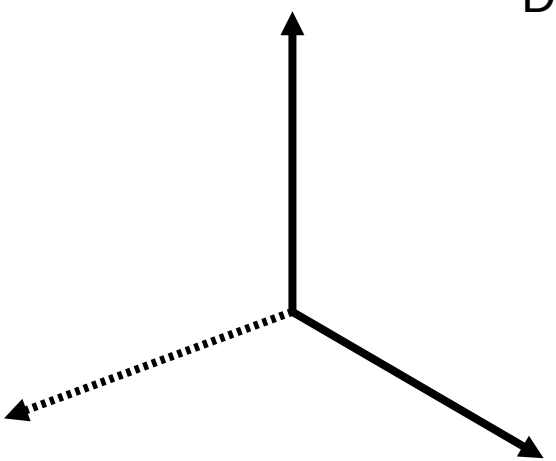
- 6 diagrammes / 13 normalisés
  1) Diagramme des cas d'utilisation
  2) Diagramme de séquence
  3) Diagramme de classe
  4) Diagramme de paquet
  5) Diagramme d'activité
  6) Diagramme état-transition

Modélisation avec UML

- Cycle en cascade (cycle obsolète mais pédagogique)



Trois vues sur la modélisation



Structurel

Diagramme de classes
Diagramme de paquets

Fonctionnel

Diagramme des cas d'utilisation

Comportemental

Diagramme d'activités
Diagramme de séquence
Diagramme d'états-transitions

Exemple fil rouge

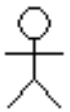
- Système simplifié de **GAB** qui offre les services suivants :
  1) Le distributeur délivre de l'argent à tout porteur d'une carte VISA.
  2) Pour les clients porteurs d'une carte de crédit de la banque, il est possible de consulter le solde de son compte, de déposer du numéraire ou des chèques.
  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 je peux faire avec le logiciel ?
- Décrire ce qu'il faut faire mais pas comment le faire

Diagramme des cas d'utilisation : syntaxe

- Acteur
  - humain ou un système que l'on développe pas



- Cas
  - Fonctionnalité du système.

- Relations
  - Include (**obligatoire**)

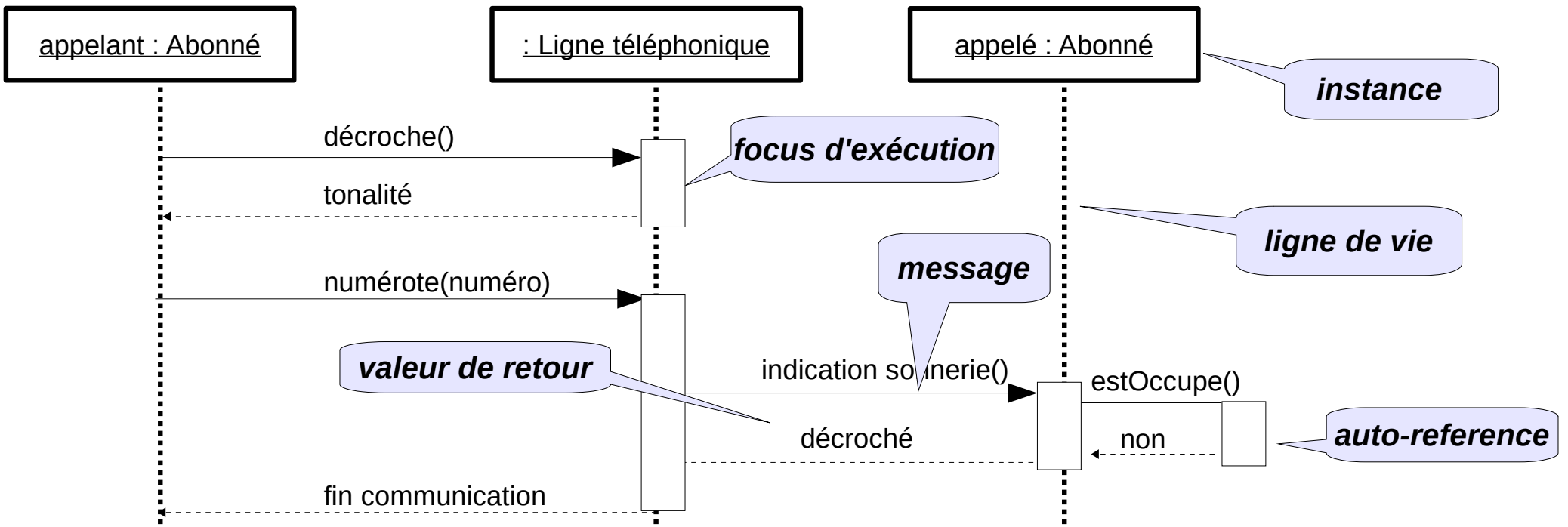
  - Extends (**facultatif**)

  - héritage

2/ Diagramme de séquence

- 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'actions entre les acteurs et le système

Diagramme de séquence : syntaxe



3/ Diagramme de classe (domaine)

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

Diagramme de classe : syntaxe

Classe

Relations :

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

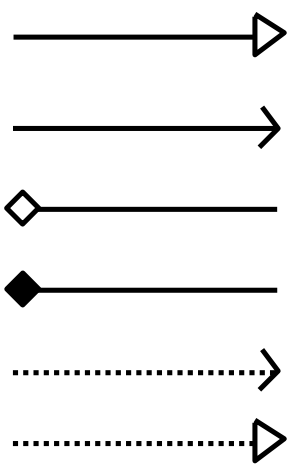
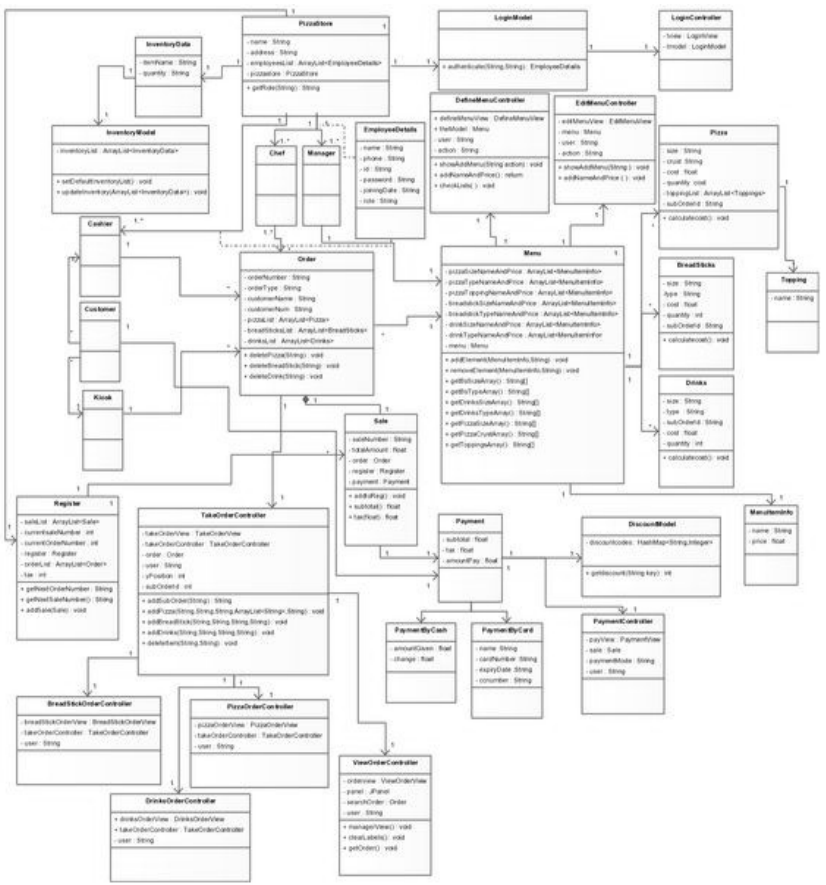


Diagramme de classes

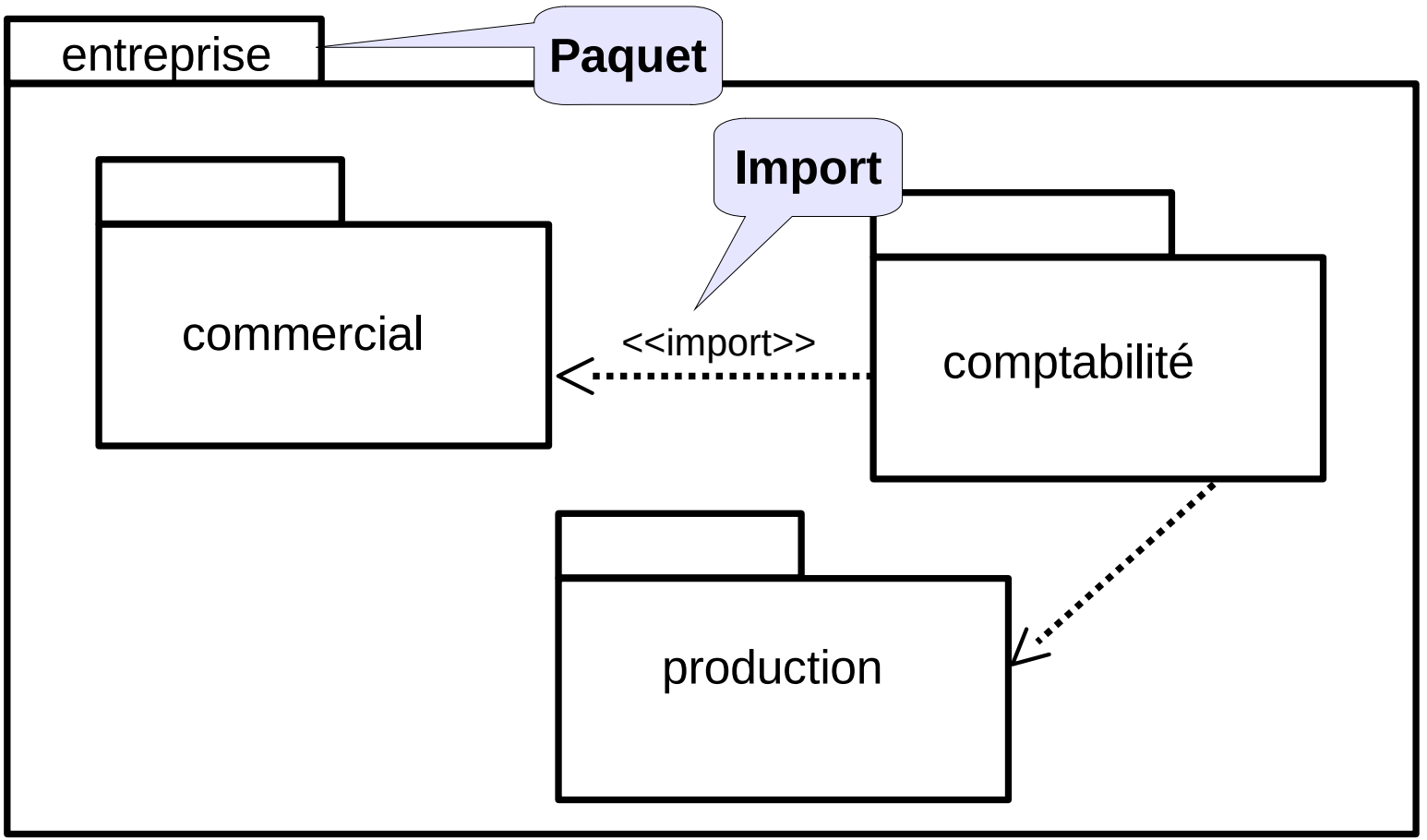
- Attention à la lisibilité ! Ne mettez pas les attributs (ni les méthodes)
- Découper en plusieurs diagrammes



4/ Diagramme de paquet

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

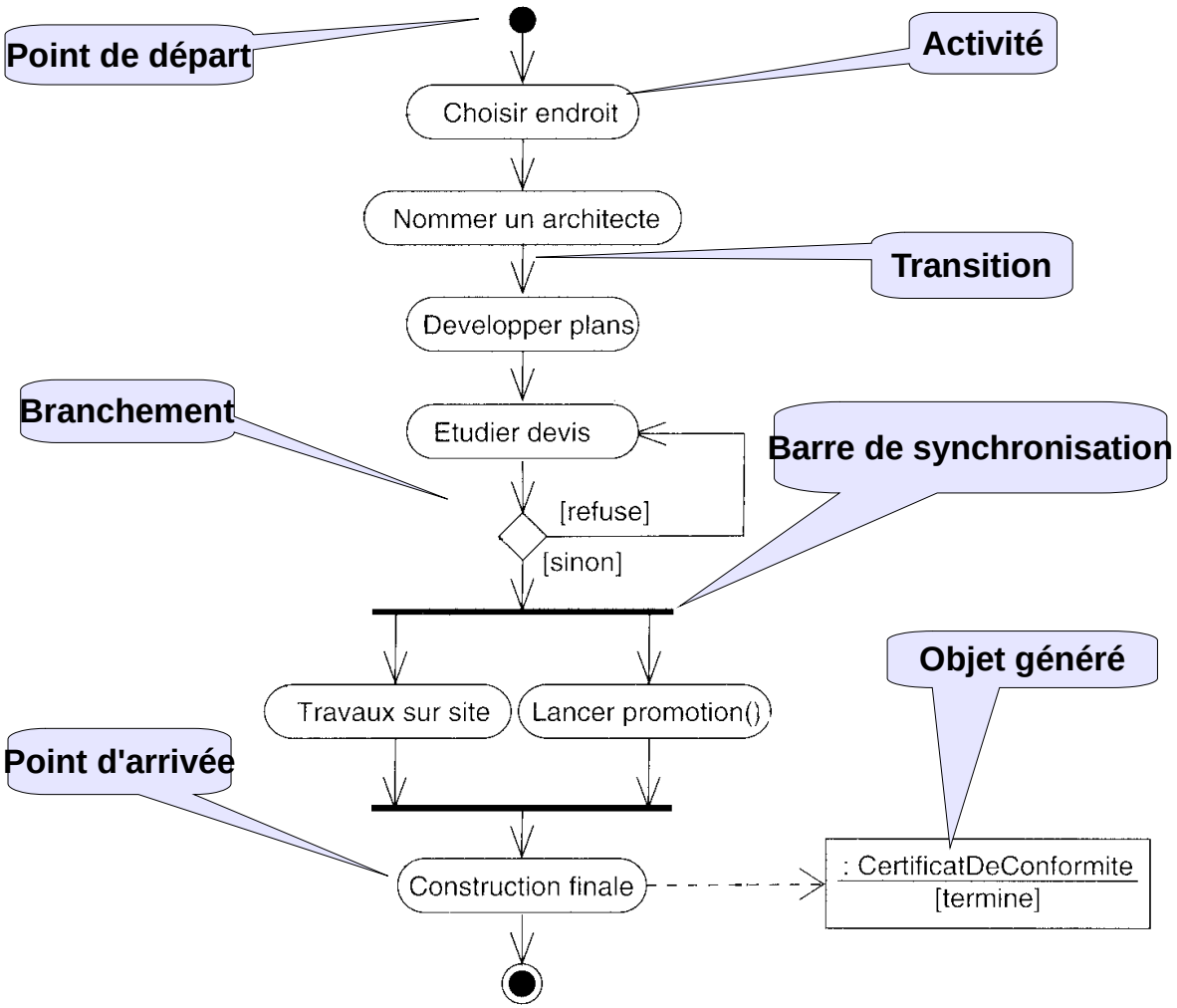
Diagramme de paquets : syntaxe



5/ Diagramme d'activité

- Modéliser un traitement
  - Un cas d'utilisation (gros grain)
  - Un algorithme pour une méthode (grain fin)
- Il est indépendant des classes

Diagramme d'activités : syntaxe



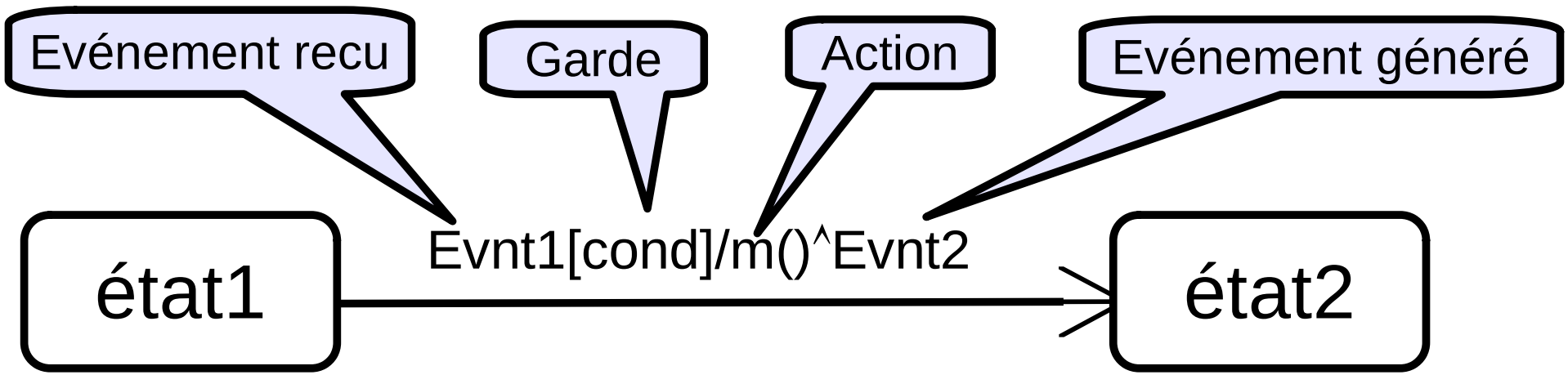
2/ Diagramme de séquence

- On peut réutiliser la diagramme de séquence pour expliciter les échanges de services entre les classes
- Niveau de granularité très fin

6/ Diagramme état-transition

- 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 logicile (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
- Les autres diagrammes sont plus spécialisés (temps réel, déploiement, infrastructure) ou des variantes de ceux présentés (communication, interaction)