Analyse et conception orientées objet

Emmanuel Polonowski

polonowski@univ-paris12.fr

Objectifs du cours

1. Analyse et conception orientées objet.

2. Modélisation avec UML 2.0.

Plan du cours

I. Introduction
   A. Processus de développement
   B. Unified Modeling Language

II. Analyse
   A. Identification des besoins : Cas d'utilisation
   B. Cas d'utilisation et Use Case Diagrams
   C. Processus métiers : Activity Diagrams

III. Conception
   A. Comportement dynamique (UML : Sequence, Communication)
   B. Visualisation des concepts (UML : Class, Package)
   C. Diagrammes UML et code Java

IV. Autres diagrammes UML

Bibliographie

- UML distilled

  M. Fowler, Addison-Wesley Pearson Education, 2004.

- UML2 et les Design Patterns

  C. Larman, Pearson Education, 2005.

- Design Patterns

  E. Gamma et al., Addison-Wesley, 1995.

Chapitre I

Introduction

Plan du chapitre I

I. Introduction

A. Processus de développement

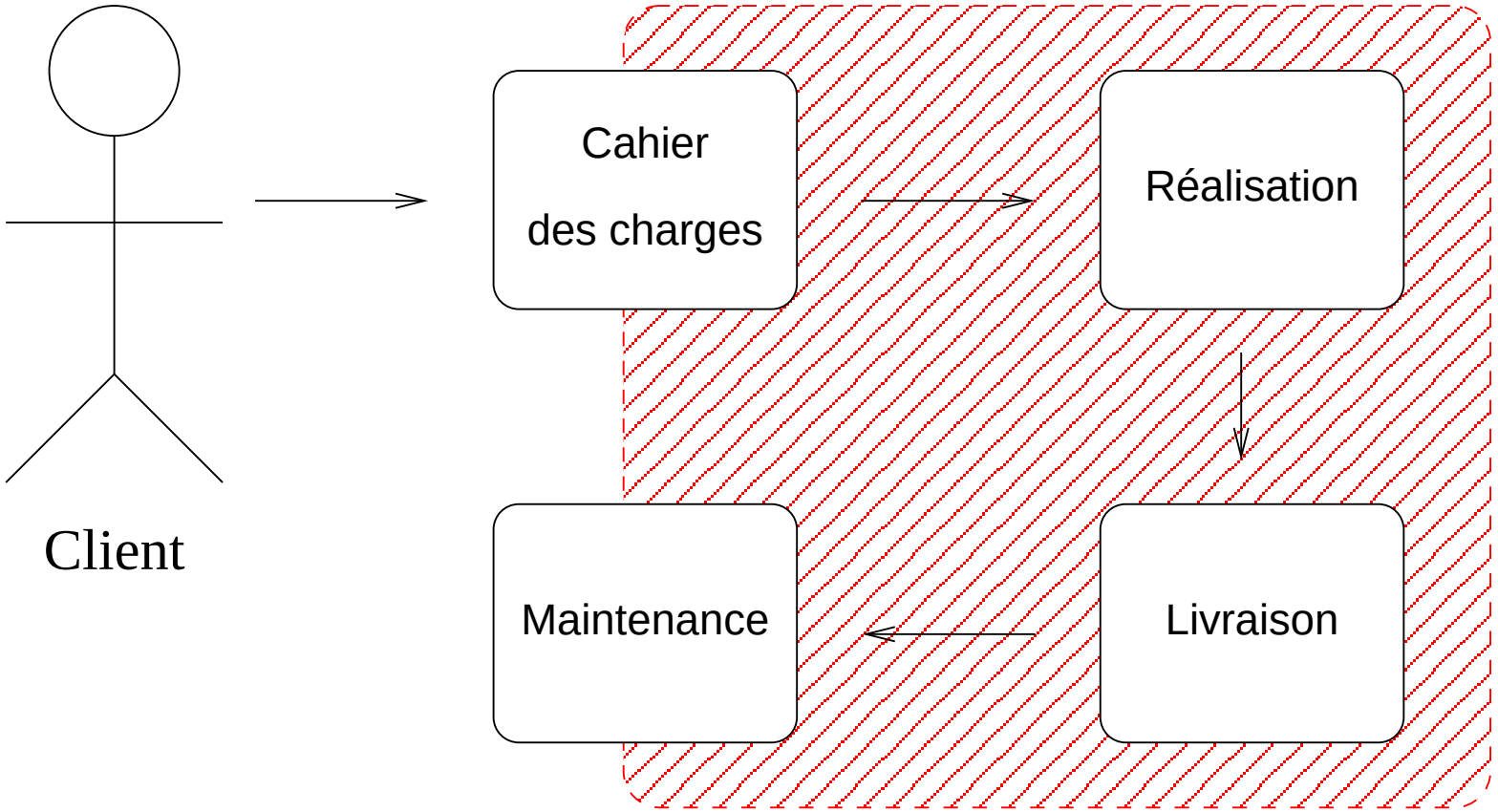
1. Fondements et cycle de vie global
2. Développement itératif ou en cascade
3. Langages de modélisation
4. Design Patterns

B. Unified Modeling Language

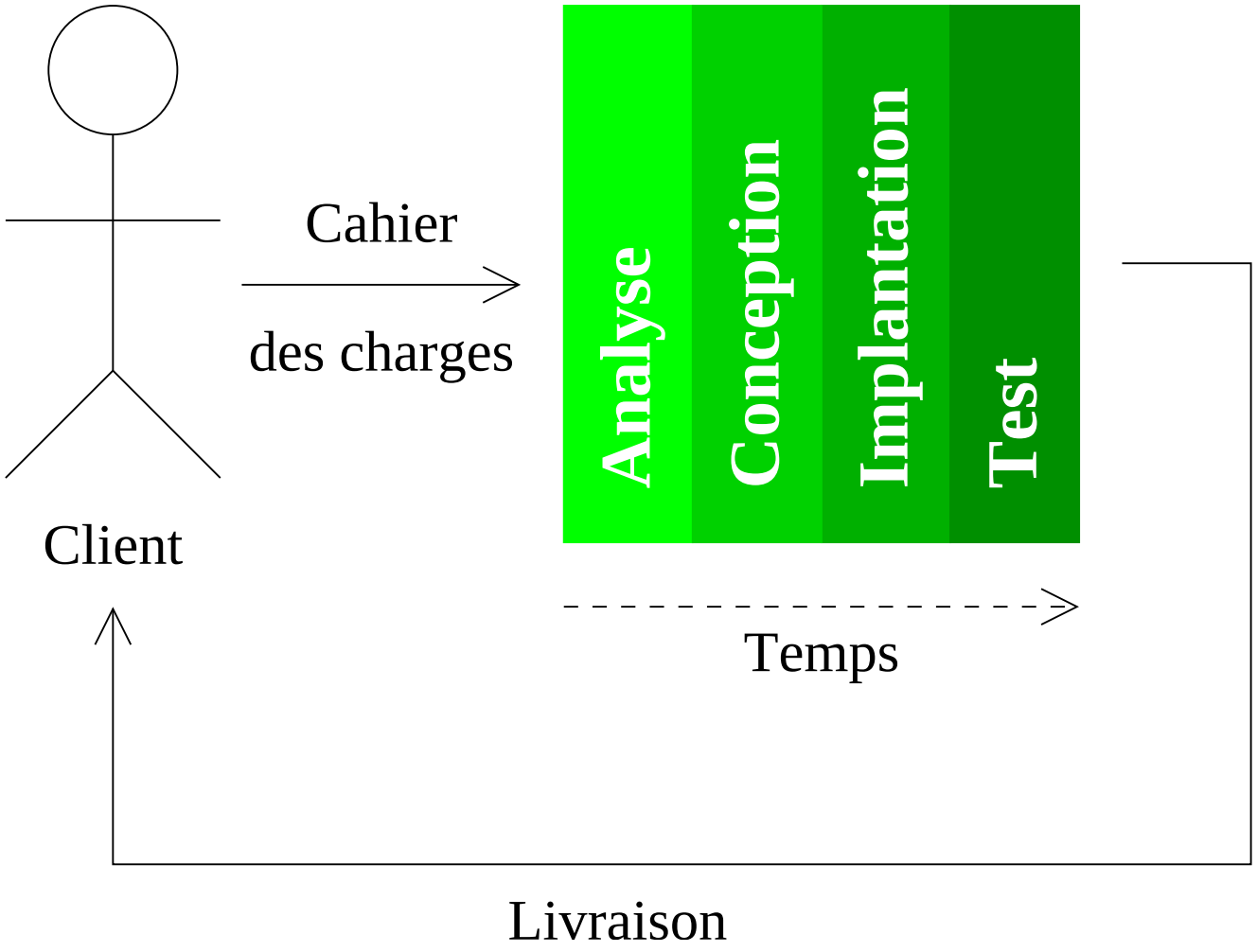
1. Historique et motivations
2. Définition et principes
3. Les diagrammes
4. Modes d'utilisation et limites
5. Utilisation des diagrammes dans le processus de développement

A. Processus de développement

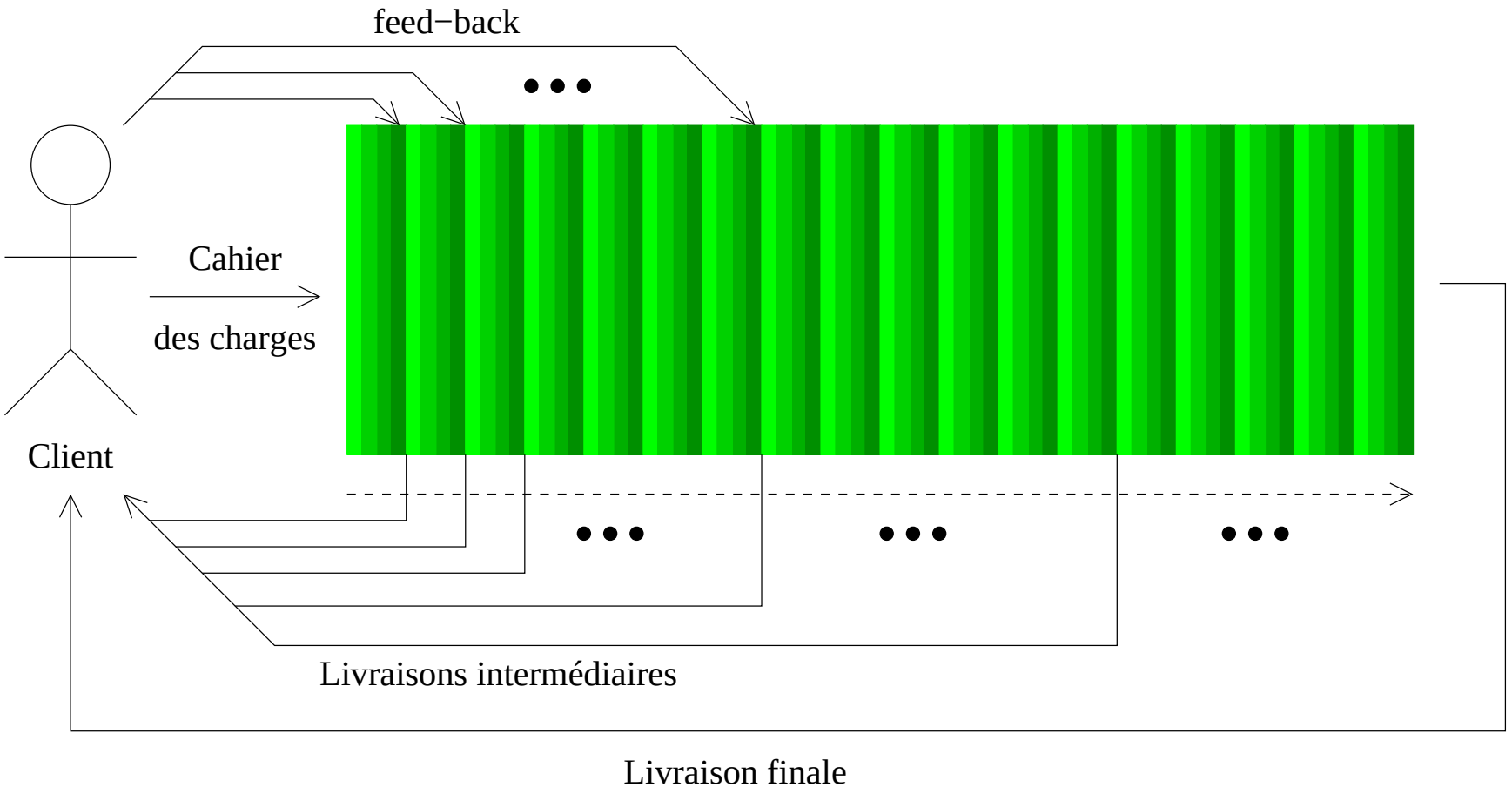
I.A.1. Généralités



I.A.2. Développement en cascade



I.A.3. Développement itératif



I.A.3. Développement itératif ou en cascade

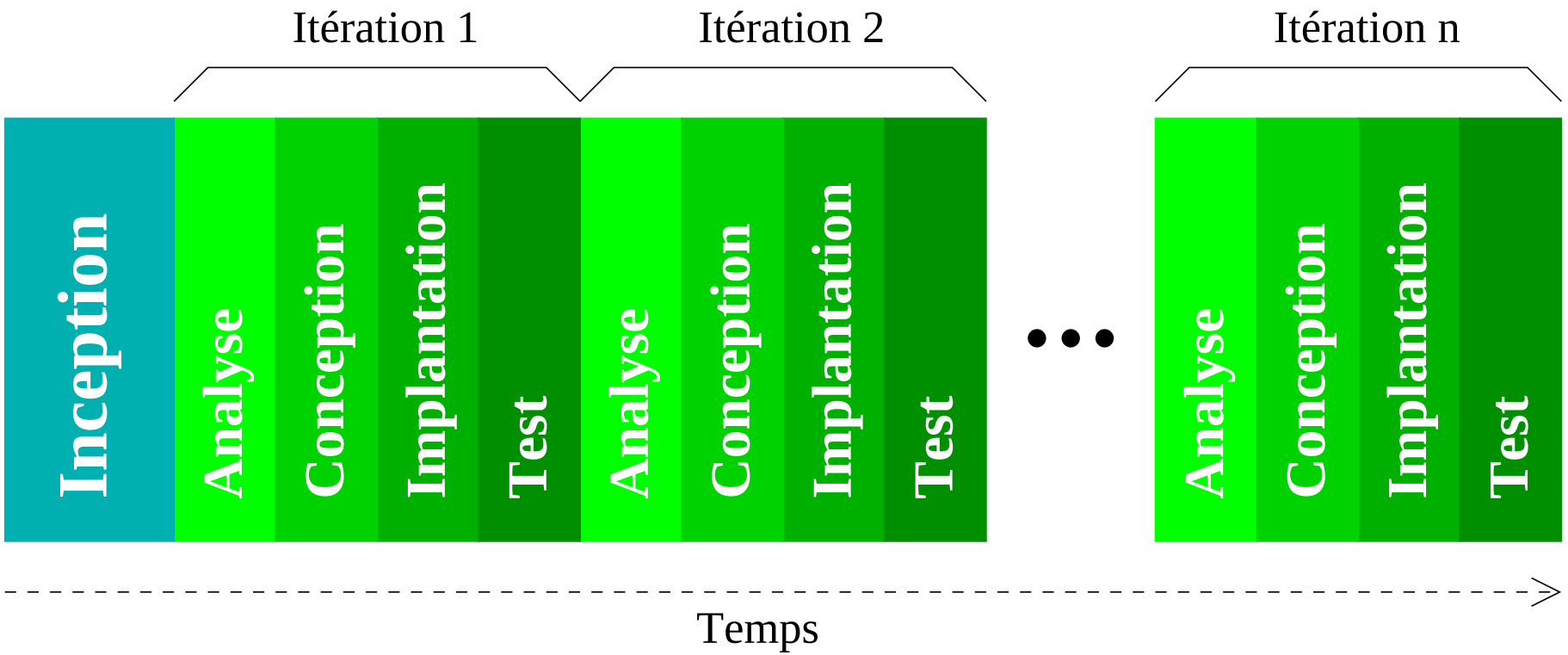
Avantages du développement en Cascade :

- Plus simple à mettre en œuvre
- Planification plus précoce

Avantages du développement Itératif :

- Diminution des échecs, amélioration de la productivité et de la qualité
- Gestion précoce des risques élevés
- Progrès immédiatement visibles
- Feed-back, implication des utilisateurs et adaptation précoces
- Complexité mieux gérée

I.A.3. Développement itératif



I.A.3. La phase d'inception

But de l'inception :

- Établir une vision commune du projet.
- Définir le périmètre de base du projet.

Pour cela :

→ Analyse détaillée d'environ 10% des cas d'utilisation :
- significatifs pour l'architecture,
- de grande valeur pour le client,
- ou à haut risque.

→ Analyse des besoins non fonctionnels critiques.

→ Réalisation de l'étude d'opportunité.

I.A.3. Principes des itérations

Buts :

- Analyser les besoins.
- Concevoir une solution.
- Implanter le code.
- Tester et livrer le produit.

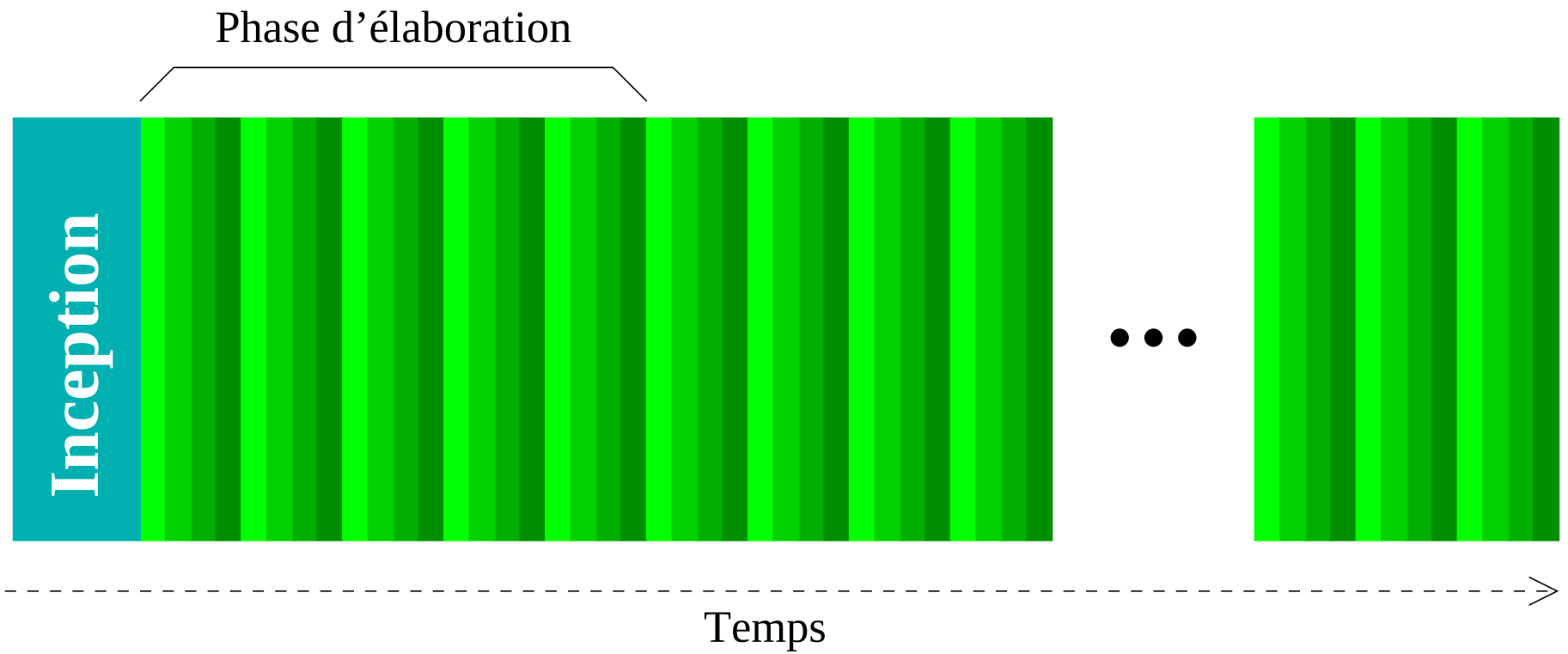
Pour cela :

→ Découpage en phases :
élaboration ⇒ construction ⇒ transition ⇒ livraison finale.

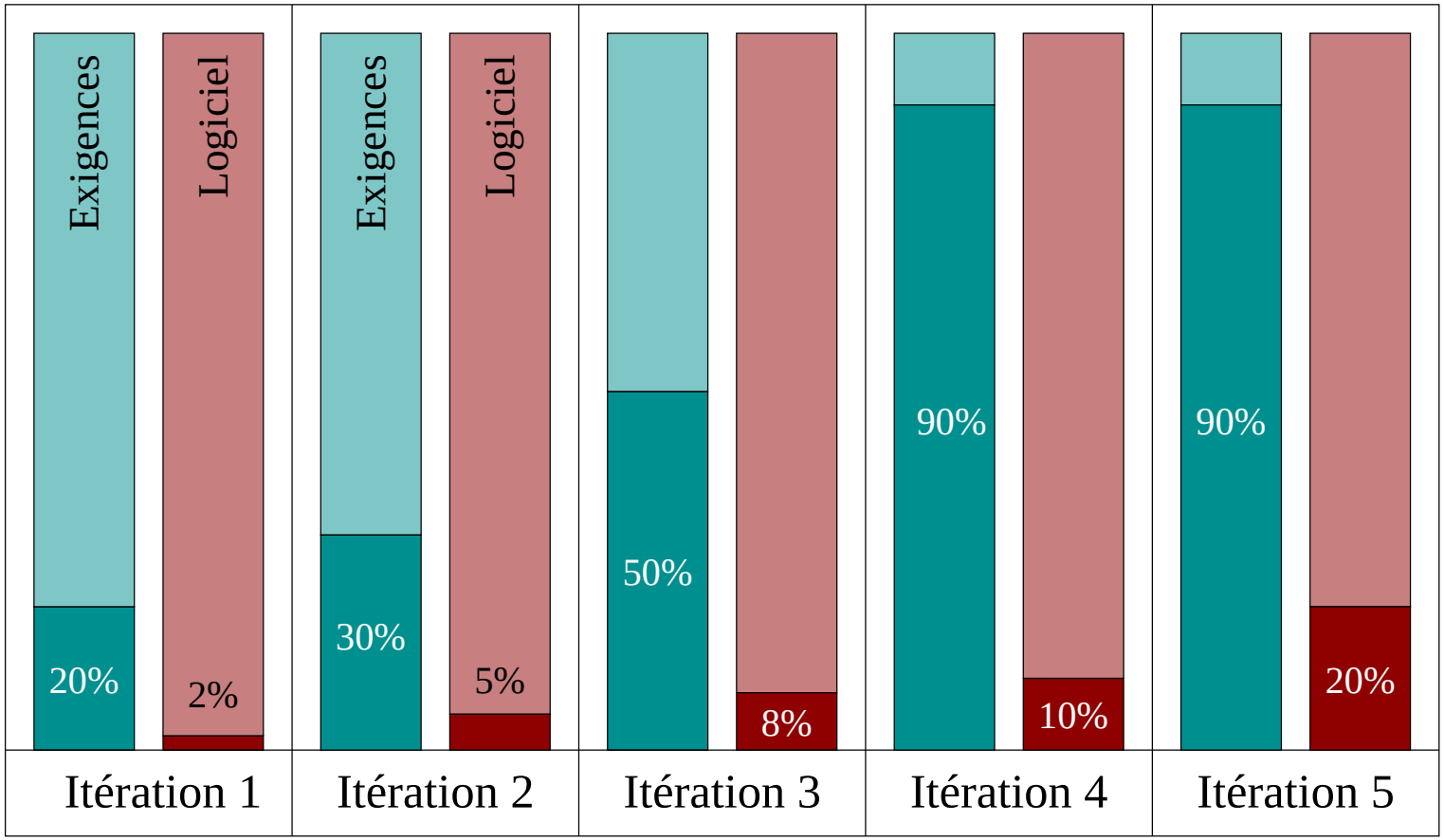
→ Augmentation itérative des fonctionnalités.

→ Livraisons successives de code opérationnel avec feedback du client.

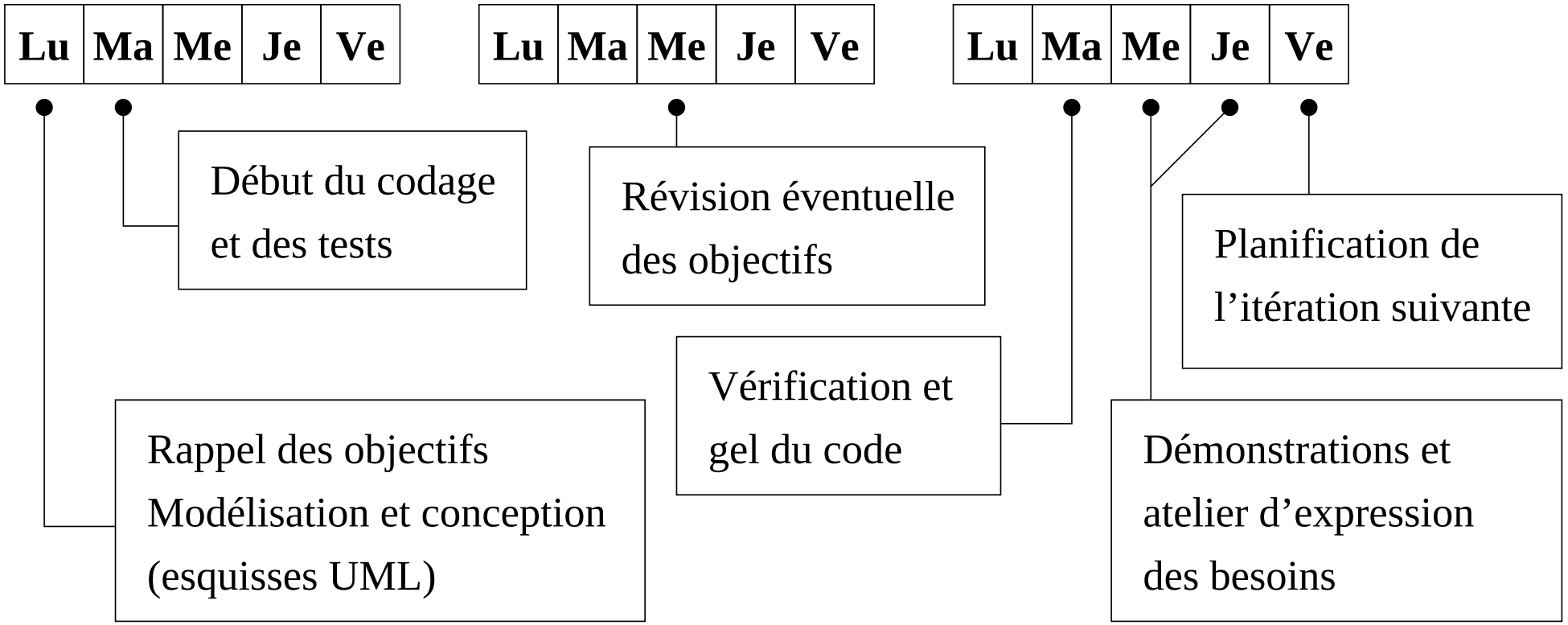
I.A.3. Développement itératif - suite



I.A.3. Développement itératif - phase d'élaboration



I.A.3. Développement itératif - Exemple d'itération



I.A.3. Analyse et Conception

Objectifs de l'analyse :

- Étude du métier du client
- Étude des besoins des utilisateurs
- Reformulation du cahier des charges sous une forme exploitable en conception

Objectifs de la conception :

- Définition de l'architecture logicielle
- Définition du comportement de l'application

I.A.4. Langages de modélisation

Intérêt :

- Formalisation des principes
- Communication des idées
- Représentation graphique

Utilisation dans le processus de développement :

- Représentation de cas d'utilisation
- Représentation du modèle du système
- Représentation des intéractions
- Représentation des classes de conception
- ...

I.A.5. Design Patterns - Patrons de conception

Intérêt :

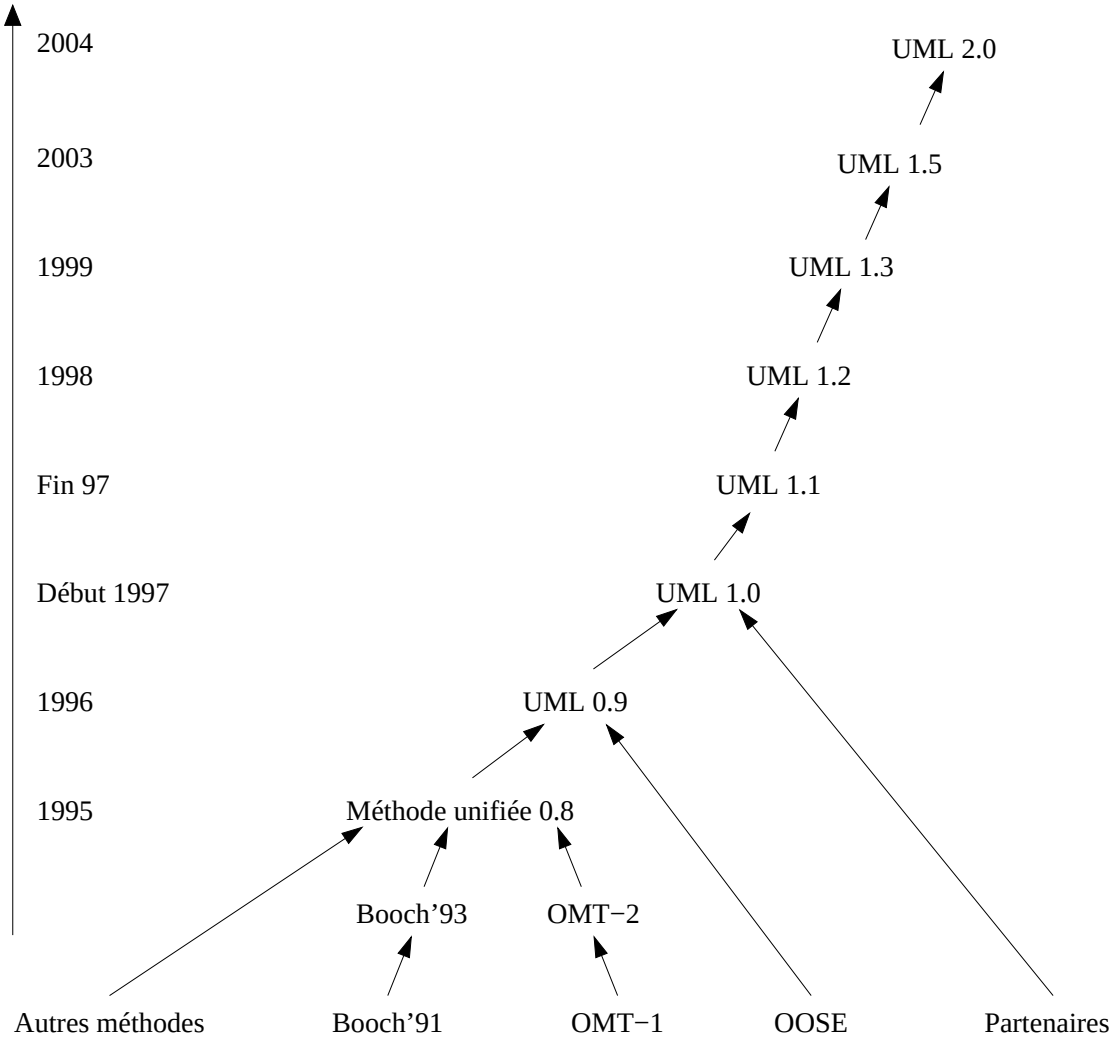
Apporter des solutions à des problèmes simples connus

Utilisation dans le processus de développement :

Pendant la modélisation et l'implantation

B. Unified Modeling Language

I.B.1 Historique et motivations



I.B.2. Définition et principes

Définition :

UML est un langage visuel dédié à la spécification,
la construction et la documentation des artefacts d'un système.

Principes :

1. Une famille de notations graphiques...
2. basée sur un méta-modèle formel...
3. permettant la génération automatique de code.

I.B.3. Les diagrammes

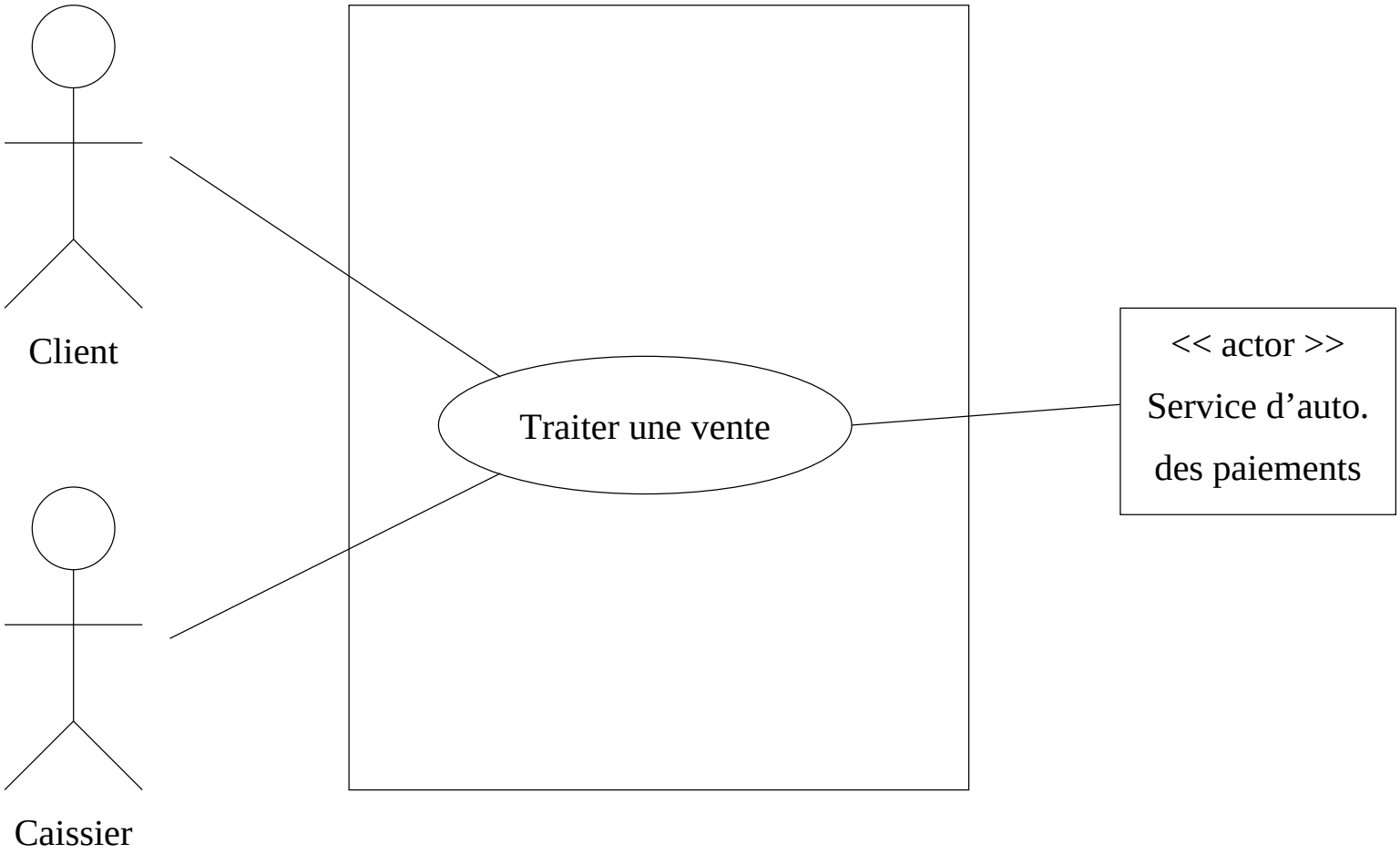
Six diagrammes de structure (ou statiques) :

• Class	• Object	• Package
• Component	• Deployment	• Composite

Sept diagrammes de comportement (ou dynamiques) :

	• Activity	• Use Case	• State Machine
Interaction Diagrams	• Sequence		• Communication
	• Interaction Overview		• Timing

I.B.3. Les diagrammes - Exemple



I.B.4. Modes d'utilisation

Esquisse :

→ Informel, dynamique, exploratoire et éphémère.

Plan :

→ Direction, processus en cascade, complet et persistant.

⇒ Nécessite une très bonne connaissance d'UML (diagrammes).

Langage de programmation :

→ Le code est généré uniquement à partir d'UML.

⇒ Nécessite une excellente connaissance d'UML (diagrammes et méta-modèle).

I.B.4. Limites

- Le langage de modélisation ne remplacera jamais la compétence de l'analyste-concepteur.

- Une bonne conception illégale en UML est préférable à une mauvaise conception légale en UML.

$\Rightarrow$ Mieux vaut chercher à se perfectionner en conception qu'en UML !

- UML n'est pas suffisant pour représenter tous les aspects utiles à la conception.

- Remarque : Dans 80% des cas, seulement 20% d'UML sont nécessaire...

I.B.5. Qualités des diagrammes

- Clarté de présentation.
- Cohérence avec le fond.
- Niveau de détail homogène.
- Clarté de présentation.

I.B.6. Utilisation des diagrammes dans le processus de développement

Inception :

- Use Case
- Activity

Itération - Analyse des besoins :

- Use Case
- Activity
- Class
- State Machine

Itération - Conception :

- Class
- Sequence
- Package
- State Machine
- Communication Diagram

I.B.7. Ordre de présentation

Importance/Utilité		Utilisation
Class	1	Use Case
Sequence	2	Activity
Package	3	Component
State Machine	4	Sequence
Activity	5	Communication
Use Case	6	Class
Object	7	Package
Communication	8	State Machine
Deployment	9	Deployment
Component	10	Object
Composite Structures	11	Composite Structures
Interaction Overview	12	Interaction Overview
Timing	13	Timing