



Modélisation objet avec UML

Pierre-Alain Muller

pa.muller@essaim.univ-mulhouse.fr

ESSAIM, 12 rue des Frères Lumière

68093 Mulhouse Cedex

Au menu

- La genèse d'UML
- Un survol d'UML
- La notation UML
- Vers un processus unifié
- La suite de l'histoire



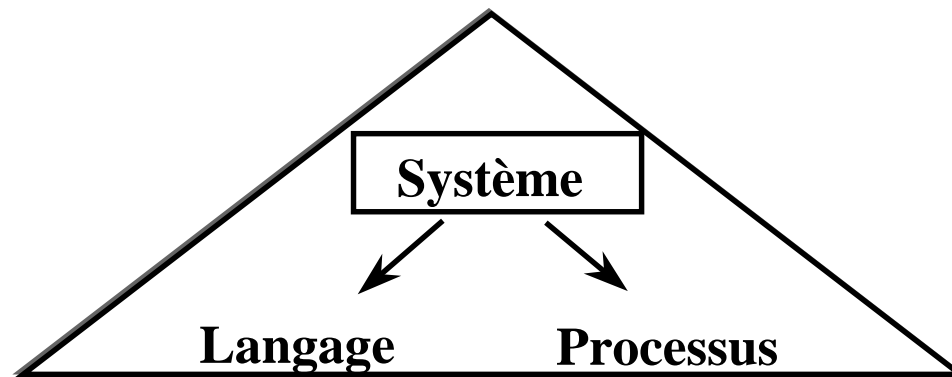
La genèse d'UML

Complexité des logiciels

- Les tendances
 - Programmation sans programmer
 - Micro-architectures (*patterns*)
 - Importance de l'architecture
 - Informatique distribuée
 - Multimédia

De quoi a-t-on besoin ?

- Un langage de modélisation
 - Notation claire
 - Sémantique précise
- Un processus de génie logiciel



Langage de modélisation

- Générique
- Expressif
- Flexible (configurable, extensible)
- Syntaxe et sémantique
- Unification par convergence aujourd'hui

Processus

- Générique
- Impossible à standardiser
 - Personnes, applications, cultures...
- Cadre configurable
- Unification par convergence dans le futur

Comment modéliser ?

- La manière de modéliser influence fortement
 - La compréhension du problème
 - La solution
- Il n'existe pas de modèle universel
 - Jeux de modèles faiblement couplés
 - Multiplicité des niveaux d'abstraction
- Les meilleurs modèles sont en prise sur le monde réel

Evolution des méthodes

- D'abord des méthodes structurées
- A partir des années 80 émergence des méthodes orientées-objets
- Les principales méthodes objets convergent
 - Différences superficielles
 - Notation, terminologie
- L'expérience permet de séparer le bon grain de l'ivraie

La prolifération des méthodes objet

- Une cinquantaine de méthodes objet dans les cinq dernières années
 - Confusion, attentisme
- Consensus autour d'idées communes
 - Objets, classes, associations, sous-systèmes, cas d'utilisation

Rapprochement de Booch et OMT

- Booch'93 et OMT-2 sont plus ressemblantes que différentes
 - Booch'93 adopte les associations, les diagrammes d'Harel, les traces d'événements
 - OMT-2 introduit les flots de messages et retire les diagrammes de flot de données
- Booch-93 construction
- OMT-2 analyse et abstraction

L'unification des méthodes

- La guerre des méthodes ne fait plus avancer la technologie des objets
- Recherche d'un langage commun unique
 - Utilisable par toutes les méthodes
 - Adapté à toutes les phases du développement
 - Compatible avec toutes les techniques de réalisation

Différentes sortes de systèmes

- Logiciels
 - Ingénierie des logiciels
- Logiciels et matériels
 - Ingénierie des systèmes
- Personnes
 - Ingénierie des affaires

Unification sur plusieurs domaines d'applications

La notation unifiée

- Basée sur les méthodes de BOOCH, OMT et OOSE
- Influencée par les bonnes idées des autres méthodes
- Mûrie par le travail en commun

Principales influences

- Souvent une histoire imbriquée

Booch	Catégories et sous-systèmes
Embley	Classes singletons et objets composites
Fusion	Description des opérations, numérotation des messages
Gamma, et al.	<i>Frameworks, patterns</i> , et notes
Harel	Automates (<i>Statecharts</i>)
Jacobson	Cas d'utilisation (<i>use cases</i>)
Meyer	Pré- et post-conditions
Odell	Classification dynamique, éclairage sur les événements
OMT	Associations
Shlaer-Mellor	Cycle de vie des objets
Wirfs-Brock	Responsabilités (CRC)

Portée de la notation unifiée

- Standardiser les artefacts du développement
 - Modèles, notation et diagrammes
- Ne pas standardiser le processus
 - Dirigé par les cas d'utilisation
 - Centré sur l'architecture
 - Itératif et incrémental

Les objectifs

- Représenter des systèmes entiers
- Etablir un couplage explicite entre les concepts et les artefacts exécutable
- Prendre en compte les facteurs d'échelle
- Créer un langage de modélisation utilisable à la fois par les humains et les machines

Approche retenue

- Identifier la sémantique des concepts de base
- Classer les concepts
- Construire un métamodèle
- Choisir une notation graphique
- Regrouper par niveau d'abstraction, complexité et domaine

Métamodèle

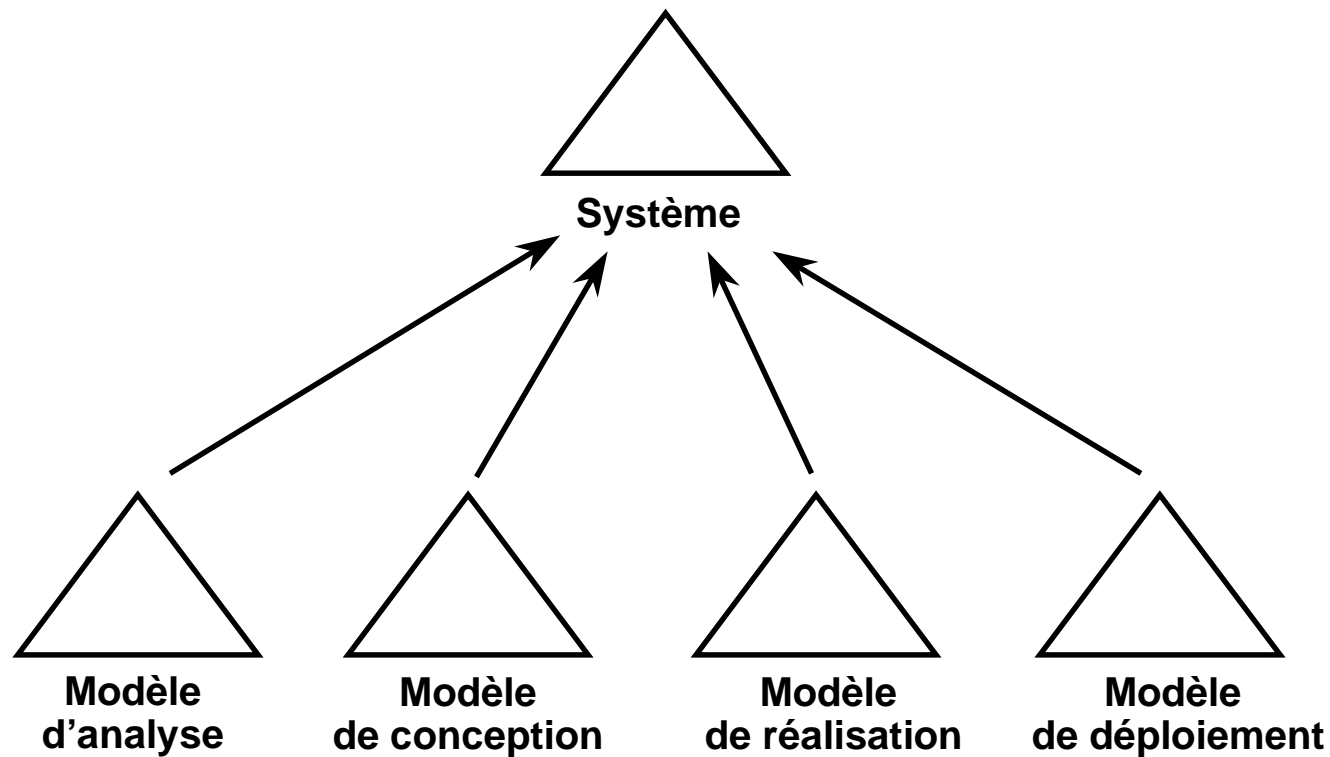
- Identification des concepts fondamentaux
 - Définition de la sémantique de ces concepts
 - Choix d'une représentation graphique
- Métamodélisation d'UML avec UML
 - Description formelle des éléments de modélisation
- Austère, pas pédagogique
 - Méthodologistes
 - Constructeurs d'outils

Les modèles et les vues

- Un modèle est un quanta de développement
 - Cohérence interne forte
 - Couplage faible avec les autres modèles
 - Relié à une phase de développement
- Une vue est une projection au travers des éléments de modélisation
 - Graphique
 - Peut englober plusieurs modèles

Exemples de modèles

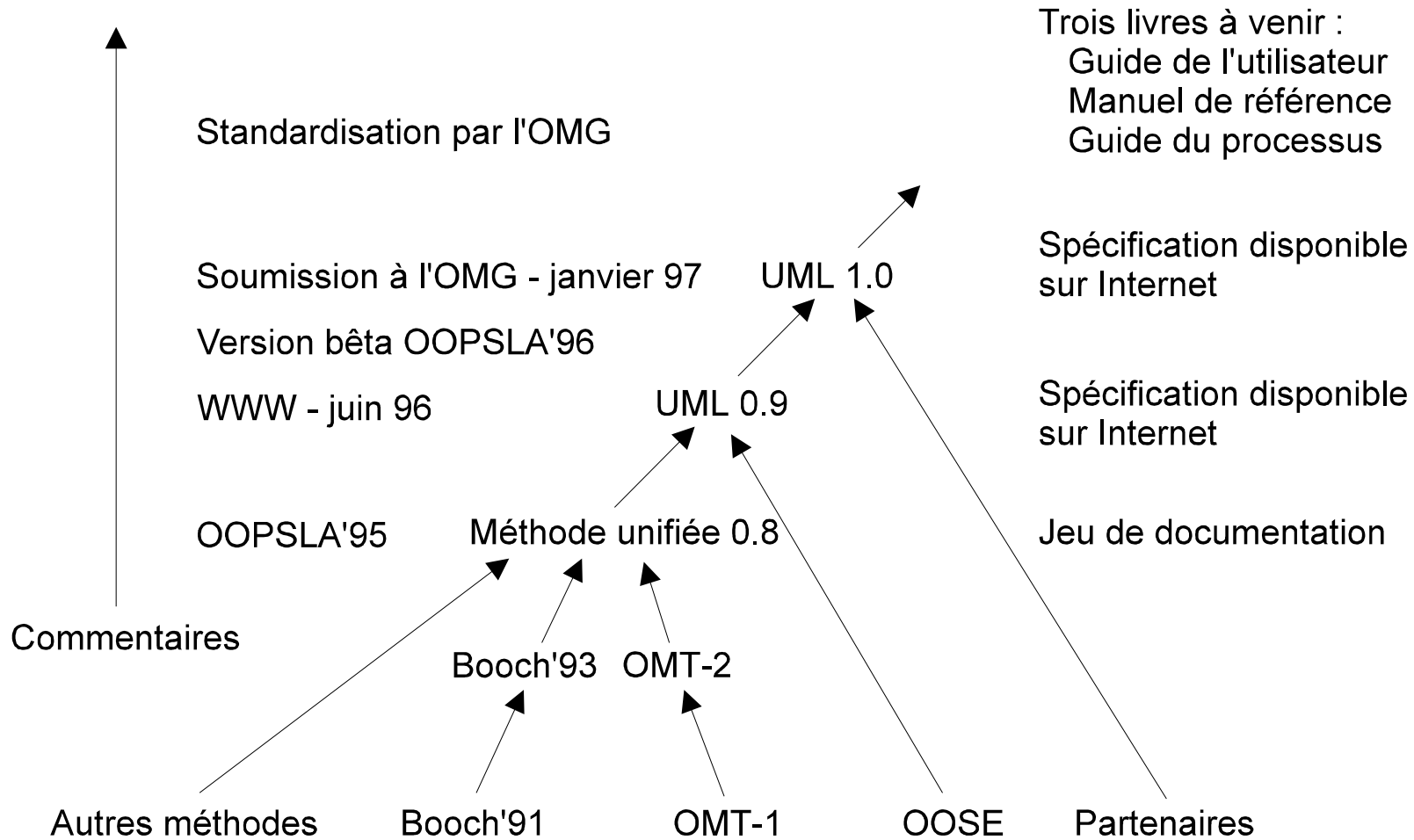
- Un système possède plusieurs modèles



Les étapes

- Octobre 95
 - *Unified Method V0.8*
- Octobre 96
 - UML V0.91 (*The Unified Modeling Language for Object-Oriented Development*)
- Janvier 97
 - UML 1.0 est soumise à l'OMG
- Septembre 97
 - Approbation par le comité technique de l'OMG

Evolution de UML



Acceptation de UML

- UML est dans le domaine public
- Soutenue par le marché
 - Microsoft, HP, IBM, Oracle...
- Successeur naturel des méthodes de Booch, OMT et OOSE
- UML est le fruit de l'expérience et des besoins de la communauté des utilisateurs

Les partenaires

- Courant 96 UML devient un enjeu stratégique
- Consortium de partenaires
 - DEC, HP, i-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI et Unisys
 - IBM, Platinum, Data Access Technologies, Reich Technologies, Softeam, Taskon A/S

En résumé

- UML est une notation, pas une méthode
- UML est un langage de modélisation objet
- UML convient pour toutes les méthodes objet
- UML est dans le domaine public







Un survol d'UML




Éléments de modélisation

- Briques pour capturer la sémantique des applications
- Pas accessibles directement aux utilisateurs
- Représentation interne (outils)
- Représentation externe (échange entre outils)

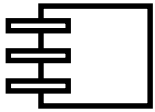
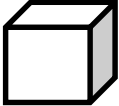


Éléments de modélisation

- Les objets 
 - Une entité d'un monde réel ou virtuel
- Les classes 
 - La description d'un ensemble d'objets
- Les états 
 - Une étape de la vie d'un objet
- Les tâches 
 - Un flot de contrôle indépendant

Éléments de modélisation

- Les cas d'utilisation 
 - Une manière dont un acteur utilise le système
- Les collaborations 
 - La réalisation d'un cas d'utilisation par une société d'objets collaborants
- Les micro-architectures (*patterns*) 
 - Un générateur pour la structure et l'interaction d'une société d'objets

Éléments de modélisation

- Les composants 
 - Un module contenant des entités d'implémentation
- Les noeuds 
 - Un dispositif matériel capable d'exécuter du logiciel
- Les paquetages 
 - Une partition du modèle
- Les notes 
 - Un commentaire, une explication ou une annotation

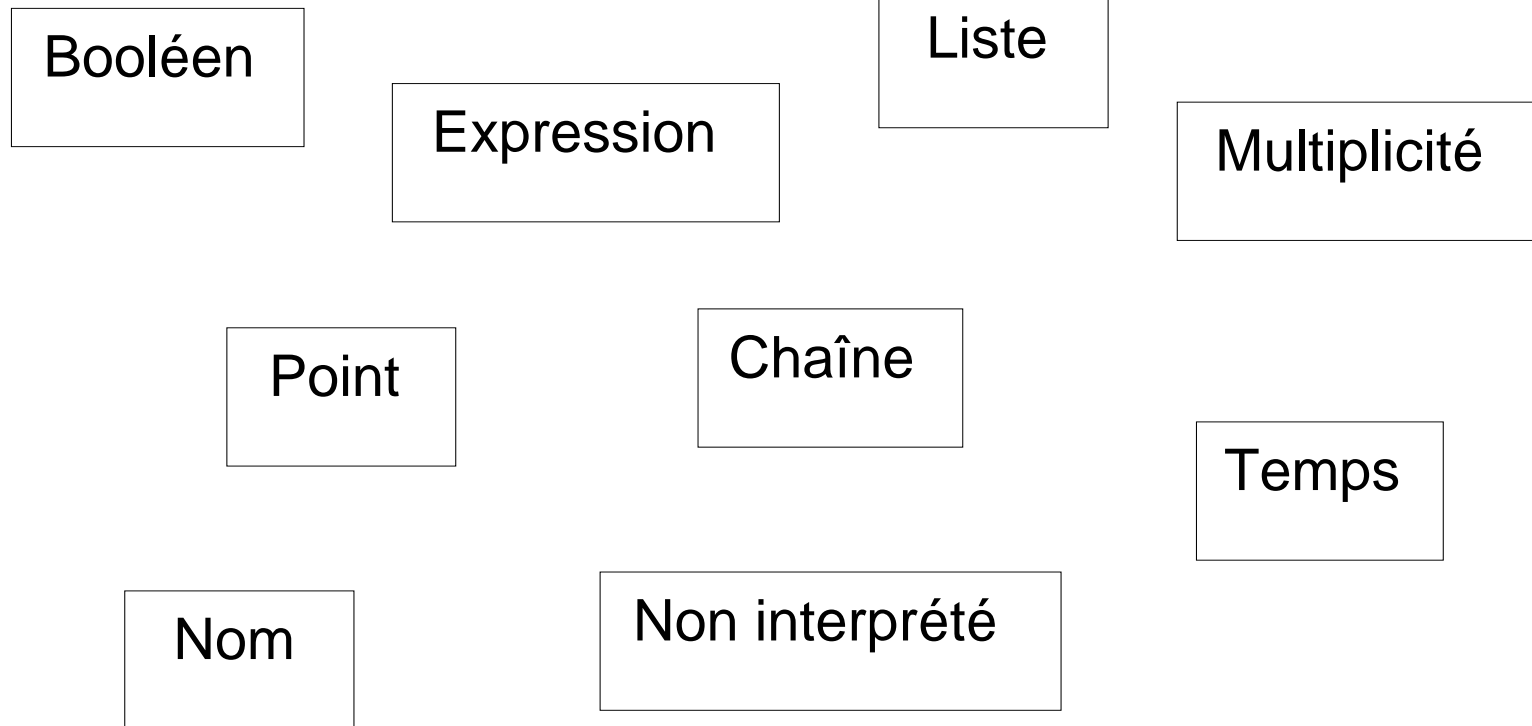
Relations

- L'association —
 - Une connexion sémantique entre instances
- La généralisation →
 - Une relation de classification
- La dépendance - - - →
 - L'utilisation d'un élément par un autre
- La trace - - - →
 - Dépendance inter-modèles

Mécanismes communs

- Les stéréotypes `<<stéréotype>>`
 - Extension des classes du métamodèle
- Les étiquettes
 - Paire (nom, valeur)
- Les notes
 - Commentaire textuel
- Les contraintes `{contrainte}`
 - Relation sémantique entre éléments

Types primitifs





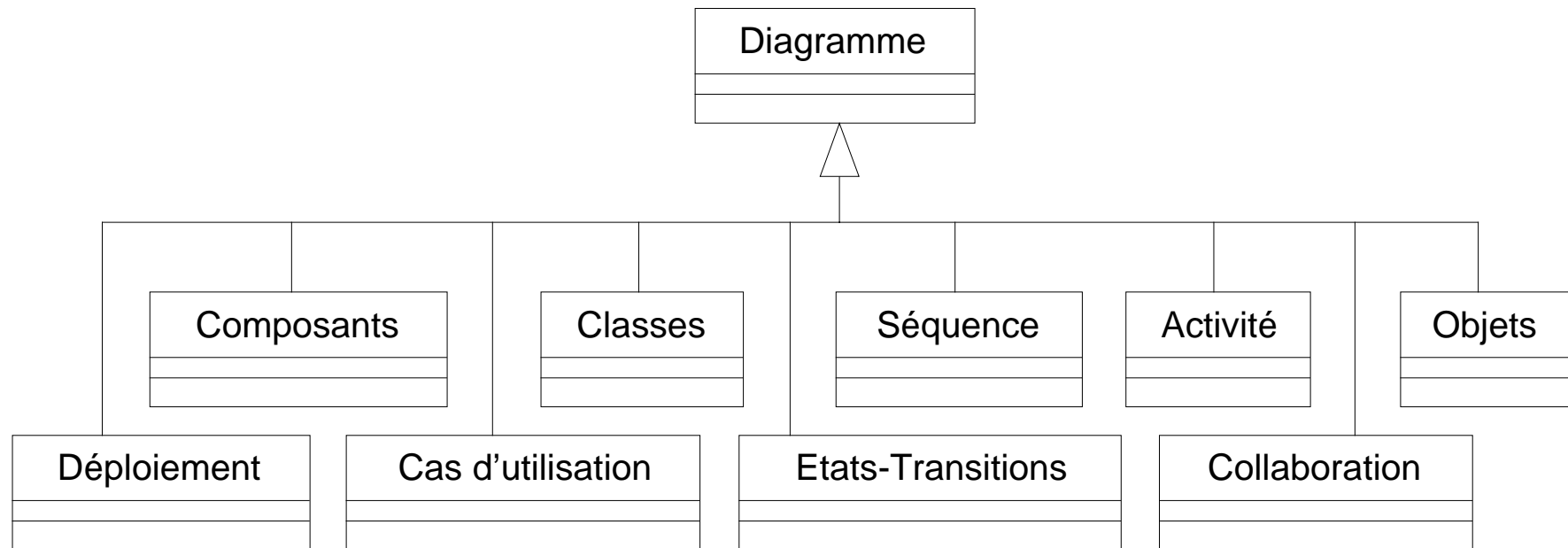
La notation UML

Notation

- Manipulée par les utilisateurs
- Simple, intuitive, expressive, cohérente
- Vues graphiques (multiples) des éléments de modélisation

Les diagrammes d'UML

- 9 types de diagrammes



Diagrammes

- Les diagrammes de classes
 - Les classes et les relations statiques
- Les diagrammes d'objets
 - Les objets et les liens
- Les diagrammes de séquence
 - Vision temporelle des interactions
- Les diagrammes de collaboration
 - Vision spatiale des interactions

Diagrammes (suite)

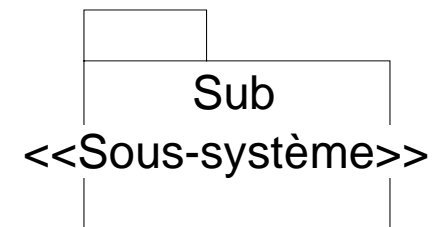
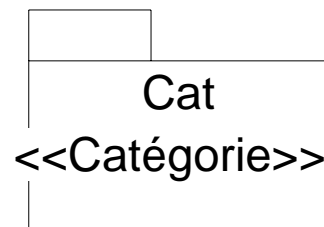
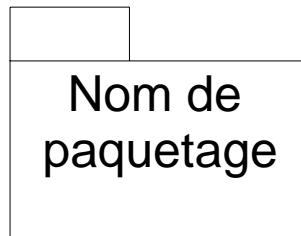
- Les diagrammes de cas d'utilisation
 - Les acteurs et l'utilisation du système
- Les diagrammes d'états-transitions
 - Le comportement des objets
- Les diagrammes d'activités
 - Le flot de contrôle interne aux opérations

Diagrammes (suite)

- Les diagrammes de composants
 - Les composants d'implémentation et leurs relations
- Les diagrammes de déploiement
 - La structure matérielle et la distribution des objets et des composants

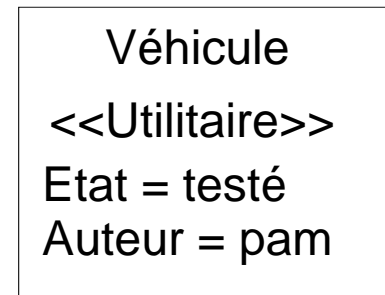
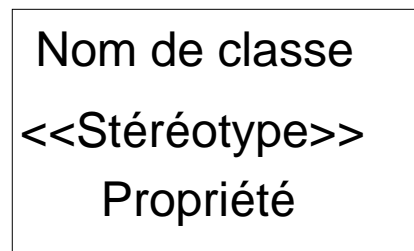
Paquetages

- Organisation des modèles



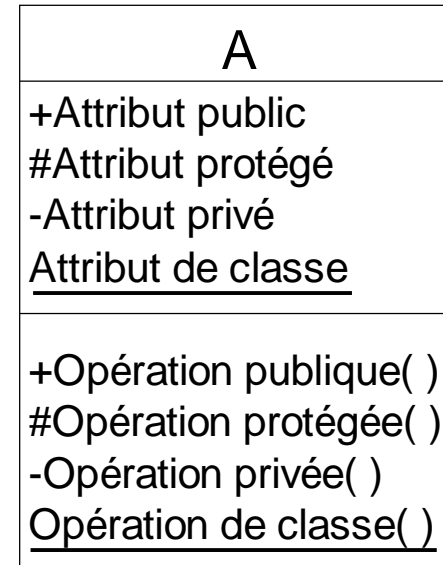
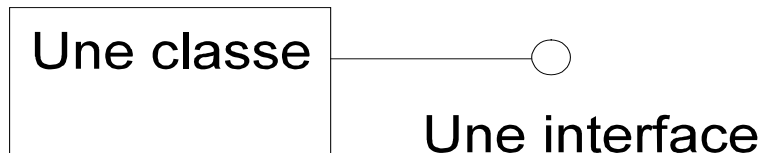
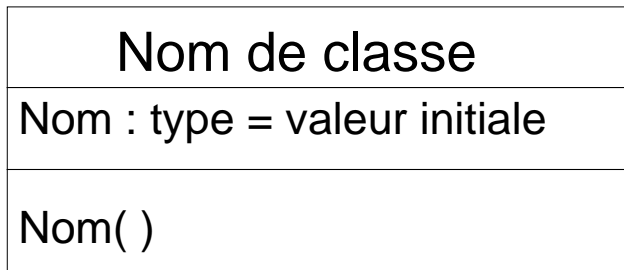
Diagrammes de classes

- Les classes



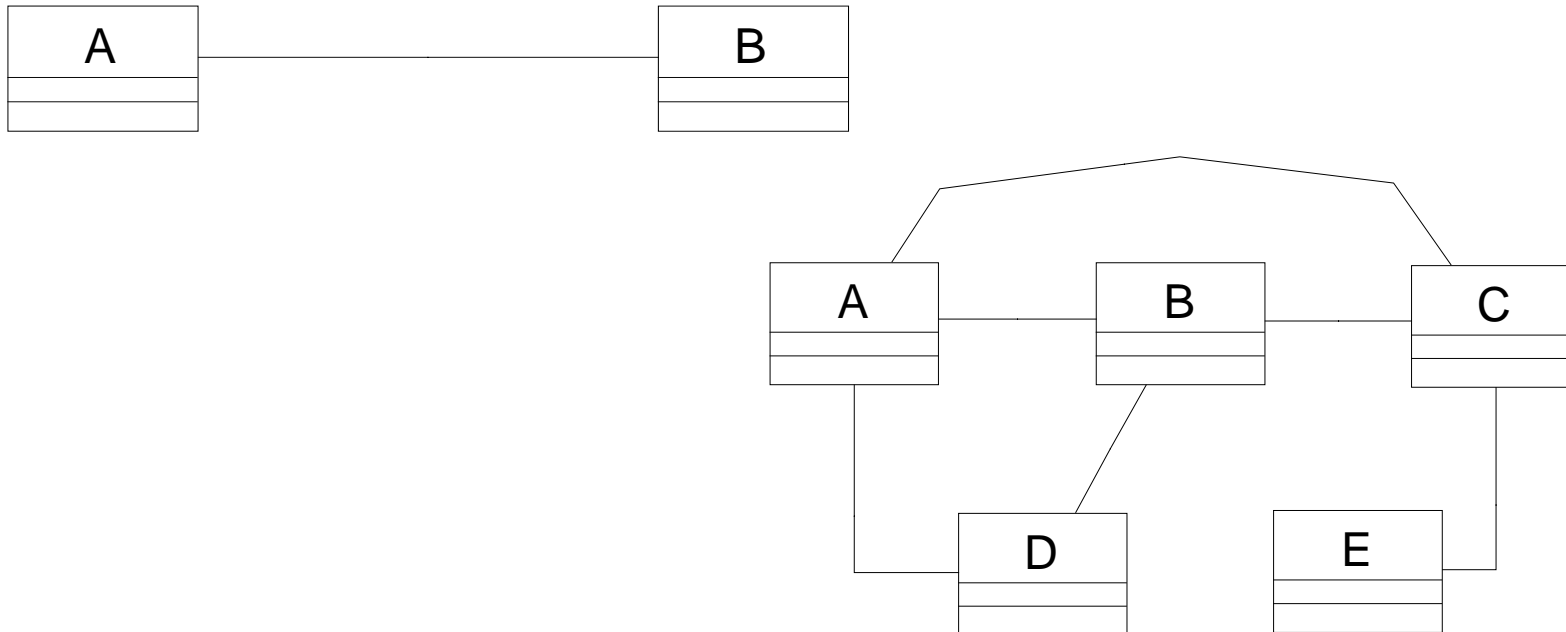
Diagrammes de classes

- Les attributs et les opérations



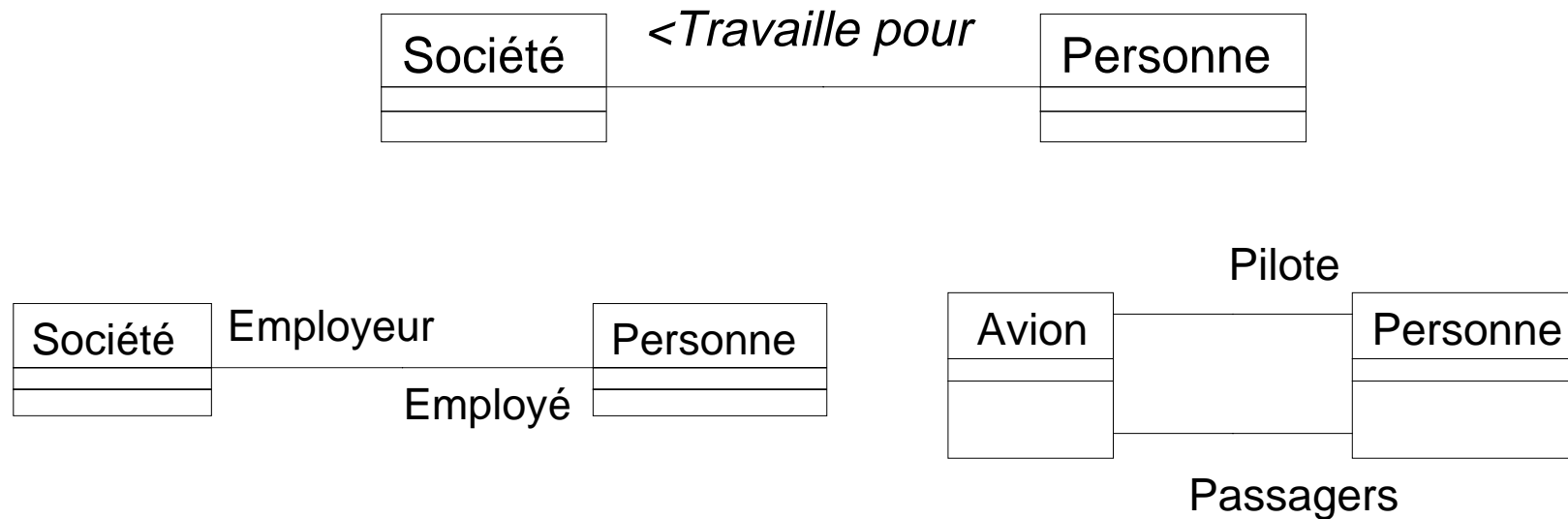
Diagrammes de classes

- Les associations



Diagrammes de classes

- Décoration des associations

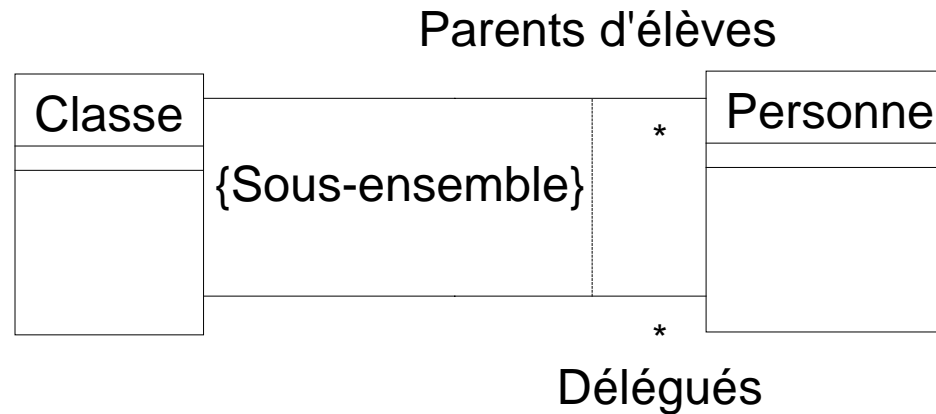
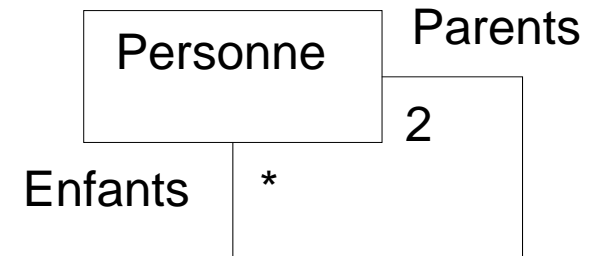
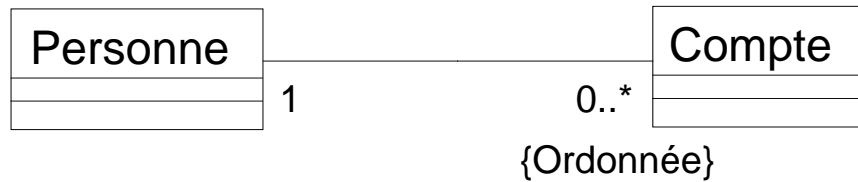


Diagrammes de classes

- Multiplicité des associations
 - 1 un et un seul
 - 0..1 zéro ou un
 - m..n de m à n
 - * de zéro à plusieurs
 - 0..* de zéro à plusieurs
 - 1..* d'un à plusieurs

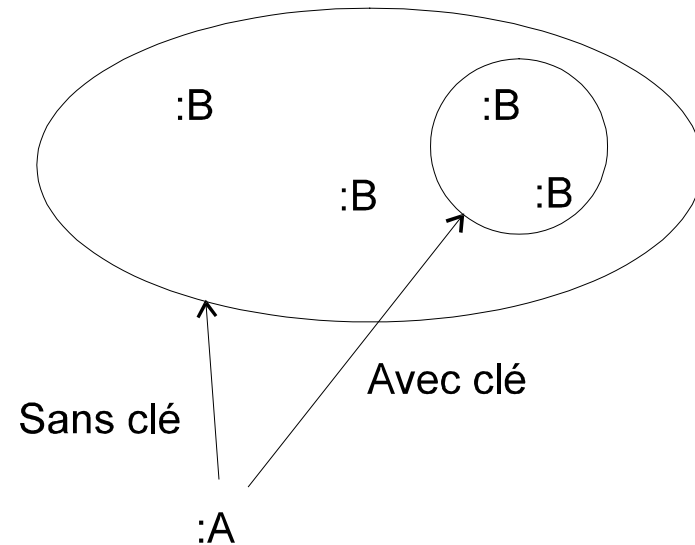
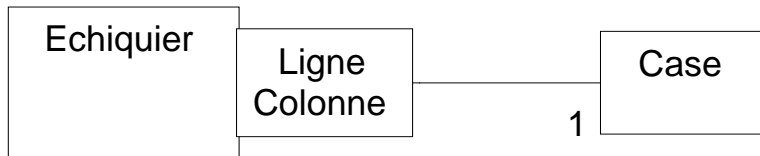
Diagrammes de classes

- Exemples



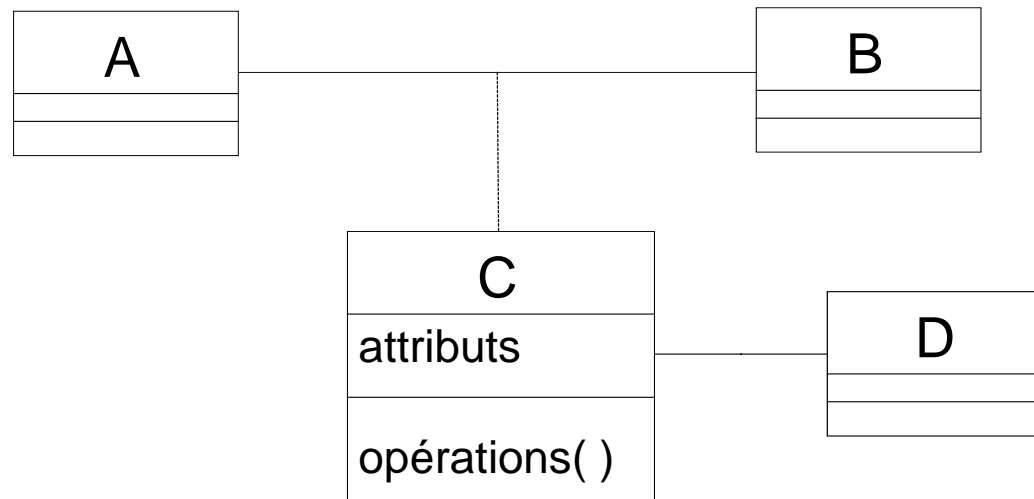
Diagrammes de classes

- Restriction des associations (*qualification*)



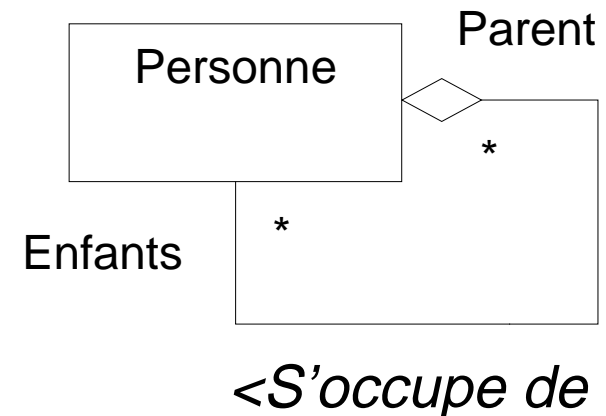
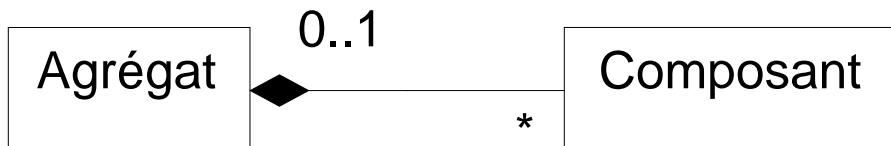
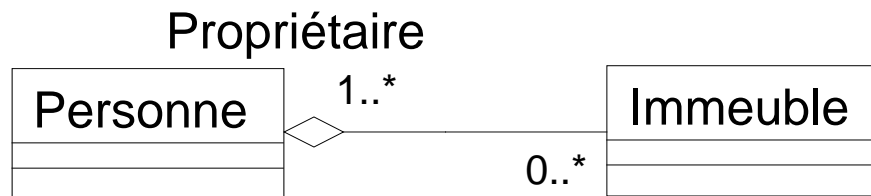
Diagrammes de classes

- Les classe-associations



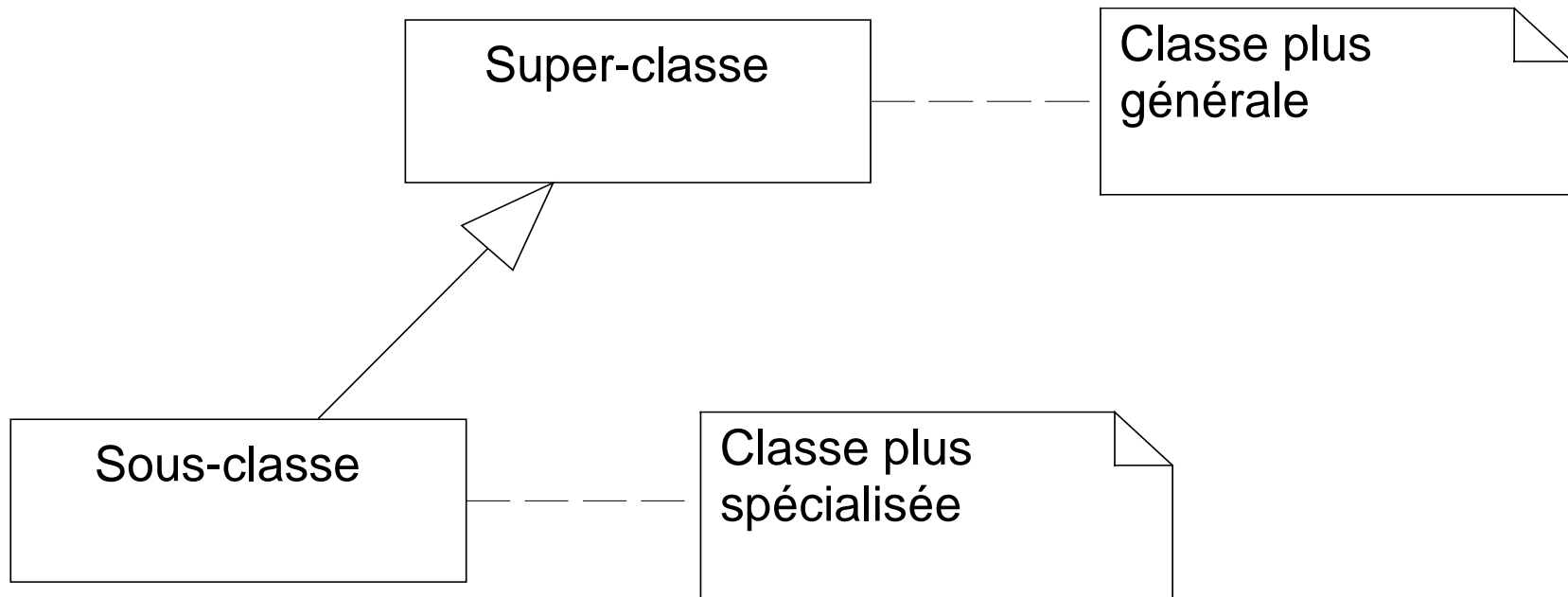
Diagrammes de classes

- Les agrégations
 - Connexions bidirectionnelles non symétriques



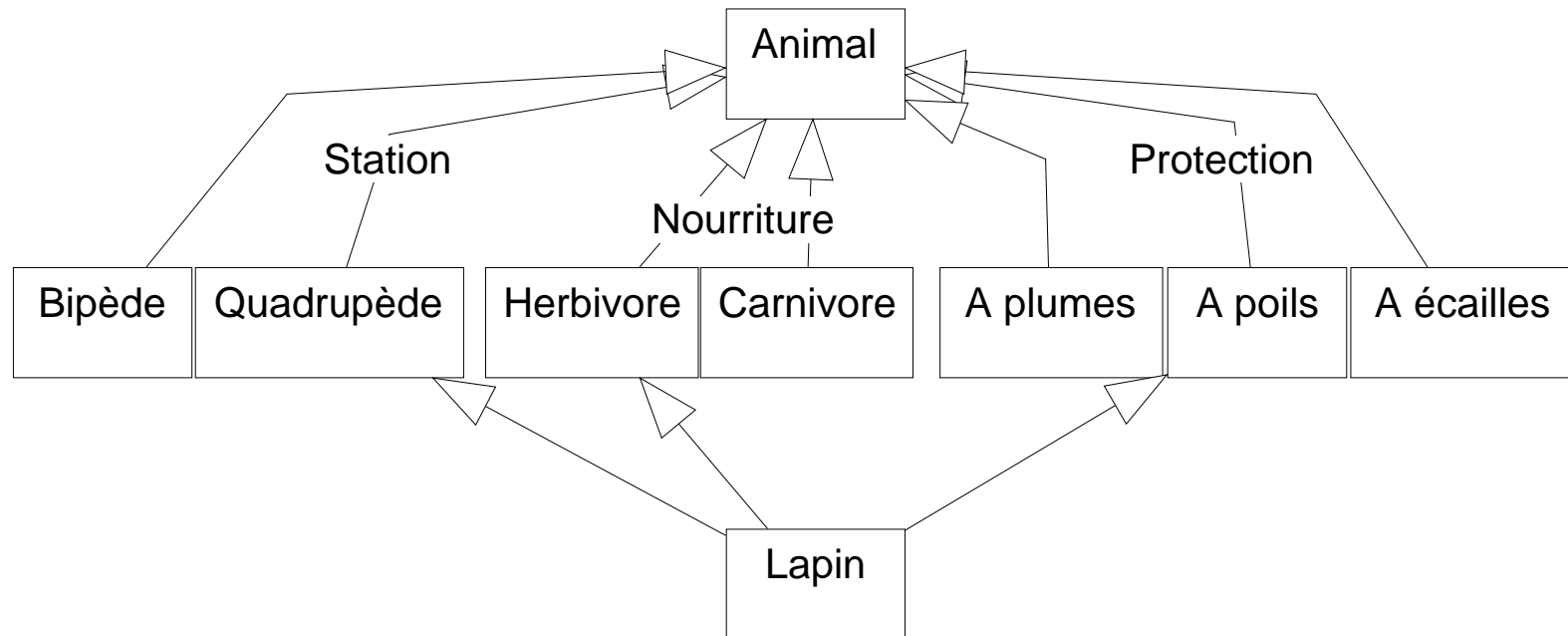
Diagrammes de classes

- Généralisation simple et multiple



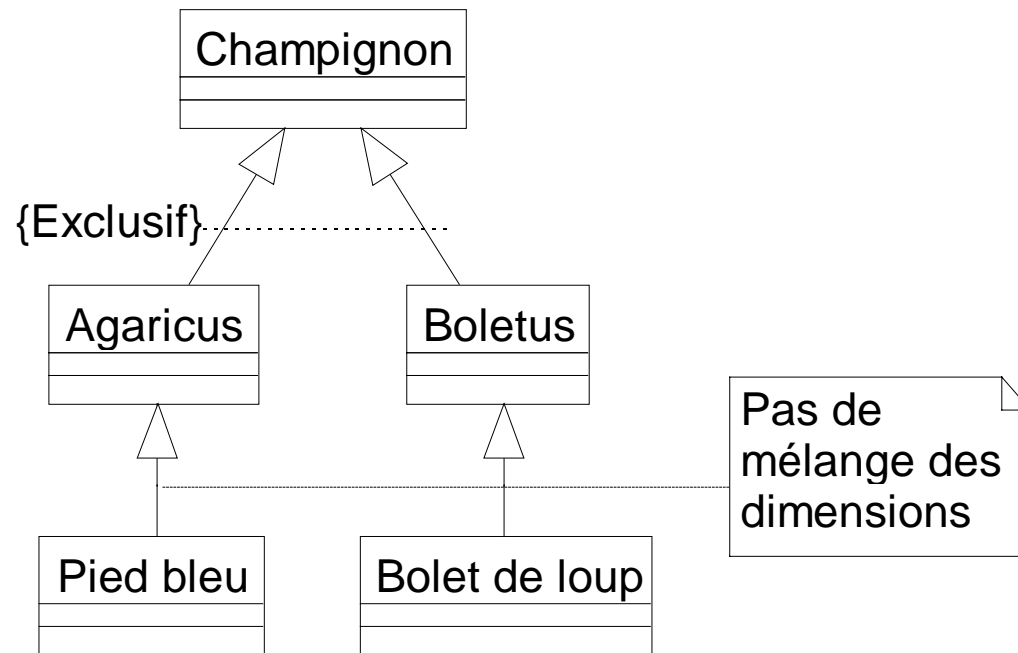
Diagrammes de classes

- Les discriminants partitionnent les sous-classes



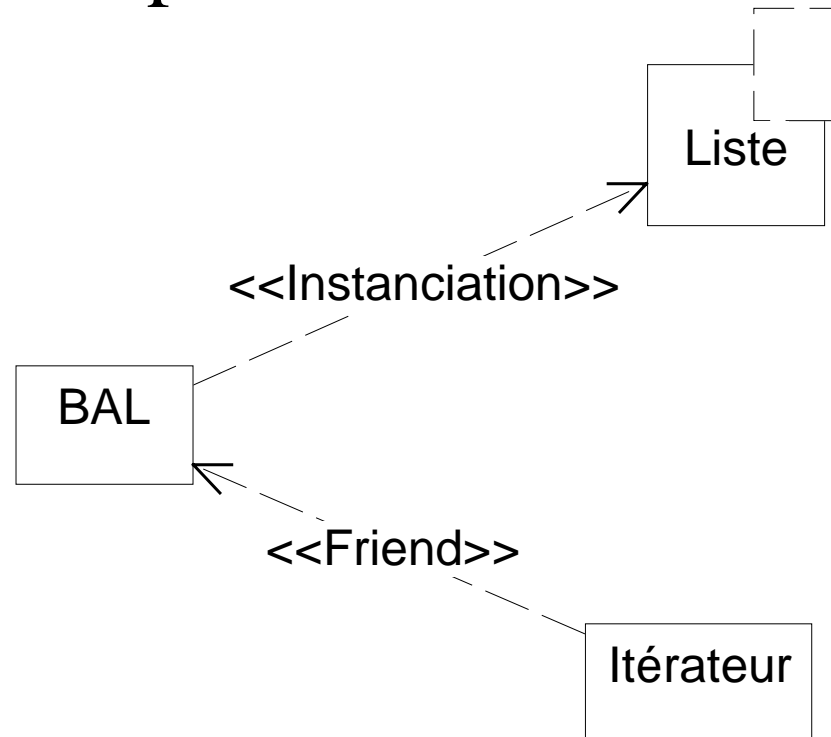
Diagrammes de classes

- Exemple de contrainte



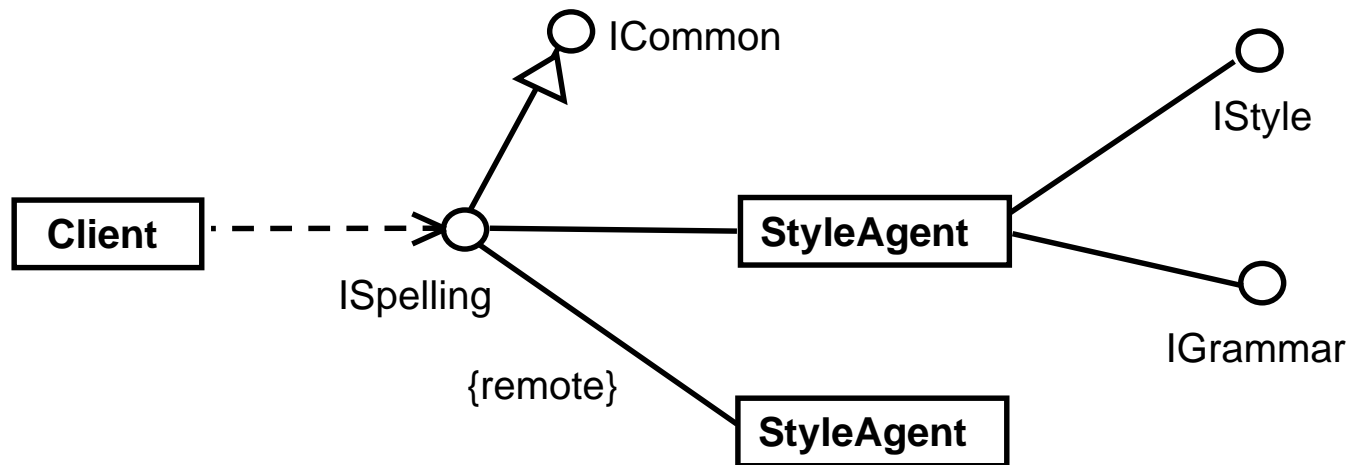
Diagrammes de classes

- Relation de dépendance



Diagrammes de classes

- Les interfaces
 - Jeu d'opérations



Diagrammes d'objets

- Représentation des objets

Nom de l'objet

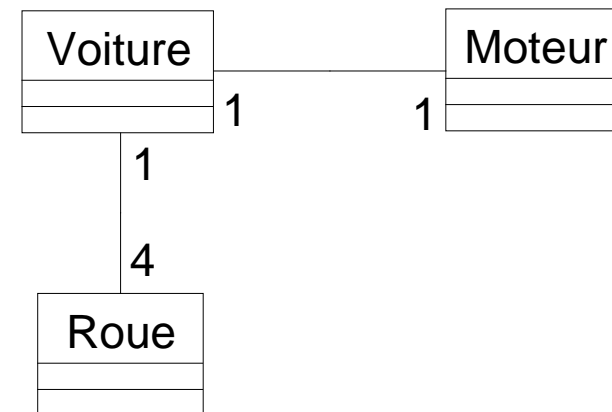
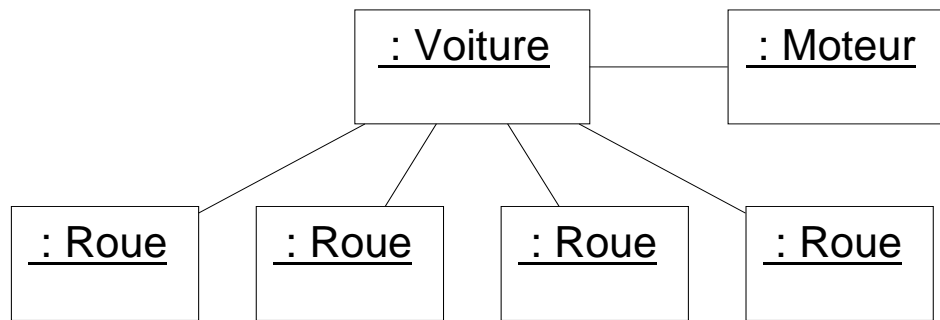
Nom de l'objet : Classe

: Classe

BoutonOK : IHM::Contrôles::BoutonPoussoir

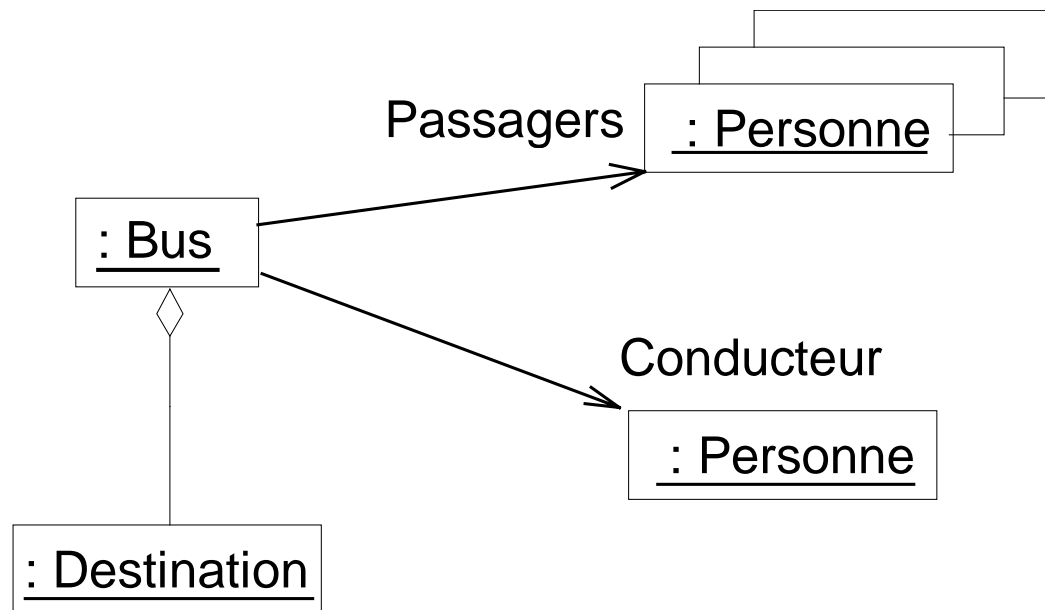
Diagrammes d'objets

- Représentation des objets et des liens



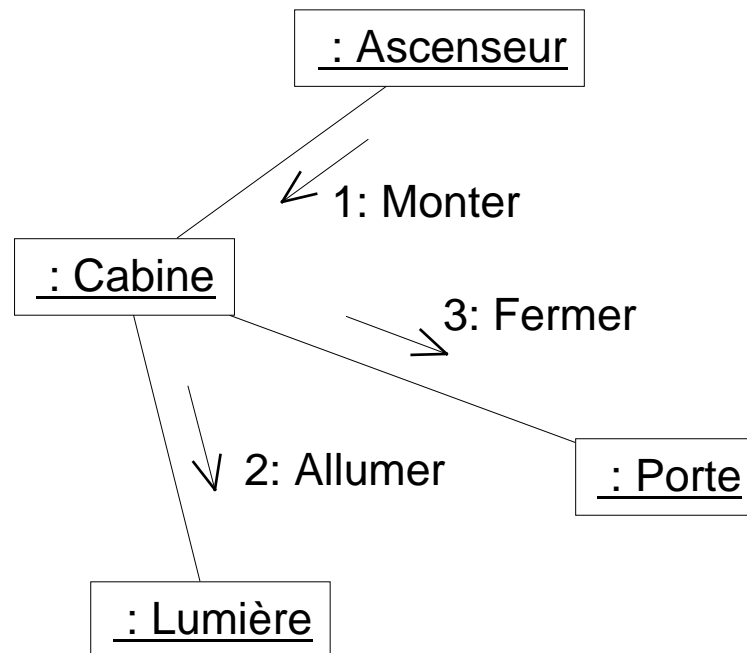
Diagrammes d'objets

- Décorations



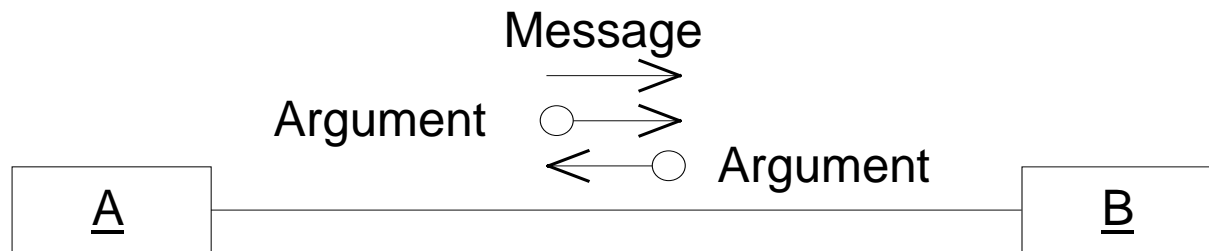
Diagrammes de collaboration

- Représentation spatiale d'une interaction



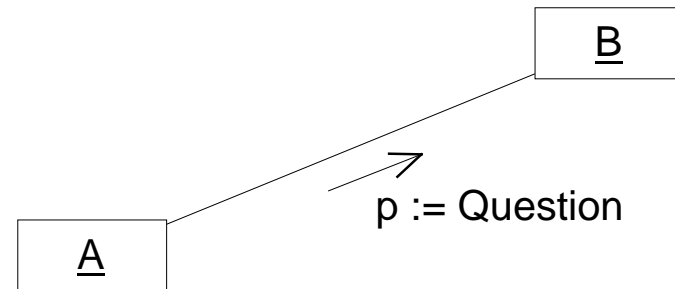
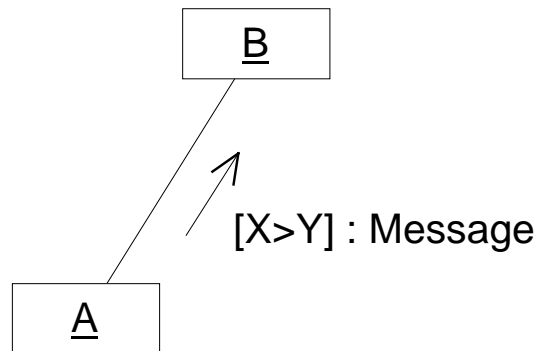
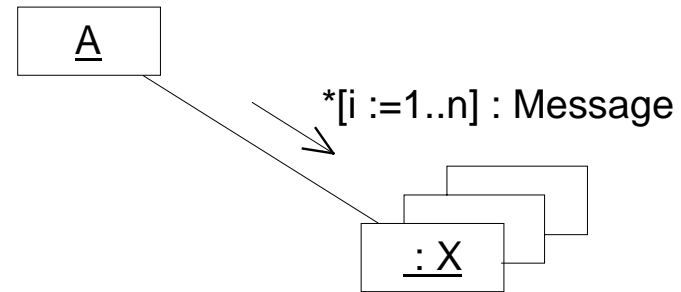
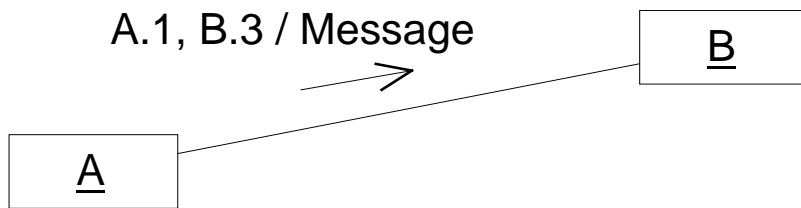
Diagrammes de collaboration

- Représentation des messages



Diagrammes de collaboration

- Exemples

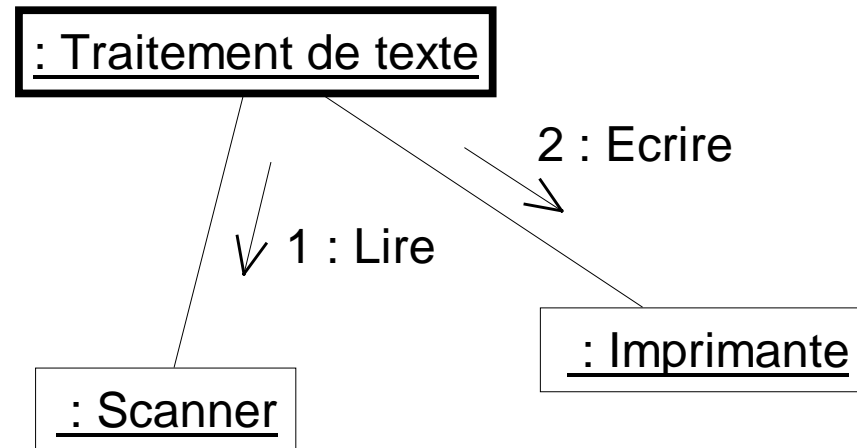


Diagrammes de collaboration

- Syntaxe des envois de message
 - 4 : Afficher (x, y) -- message simple
 - 3.3.1 : Afficher (x, y) -- message imbriqué
 - 4.2 : âge := Soustraire (Aujourd'hui, DateDeNaissance) -- message imbriqué avec valeur retournée
 - [Age >= 18 ans] 6.2 : Voter () -- message conditionnel
 - 4.a, b.6 / c.1 : Allumer (Lampe) -- synchronisation avec d'autres flots d'exécution
 - 1 * : Laver () -- itération
 - 3.a, 3.b / 4 *||[i := 1..n] : Eteindre () -- itération parallèle

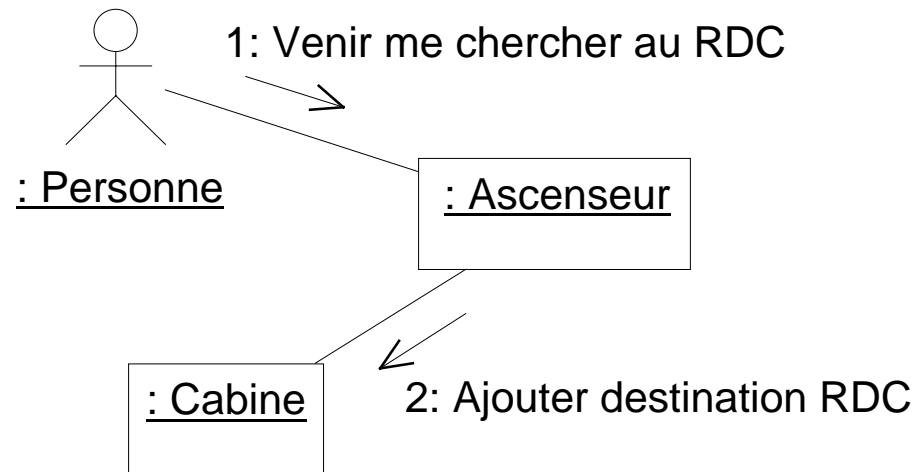
Diagrammes de collaboration

- Les objets actifs



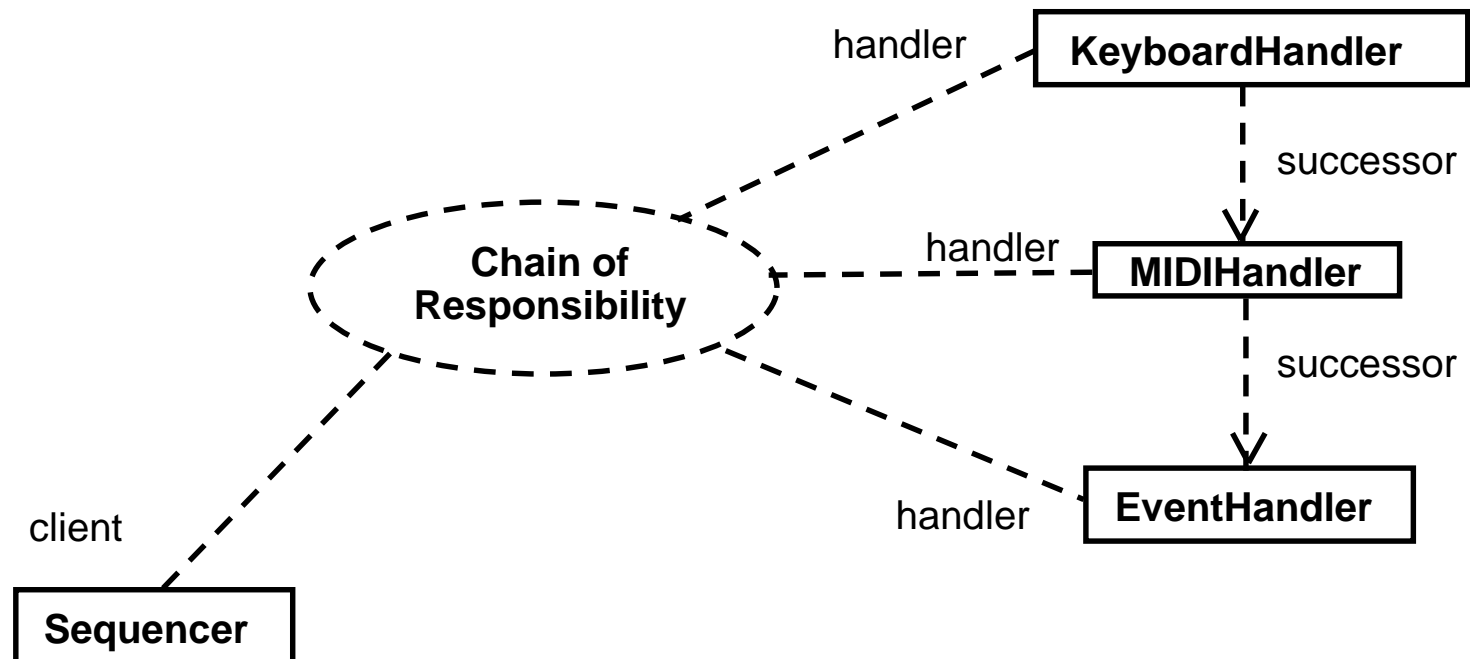
Diagrammes de collaboration

- La place de l'utilisateur



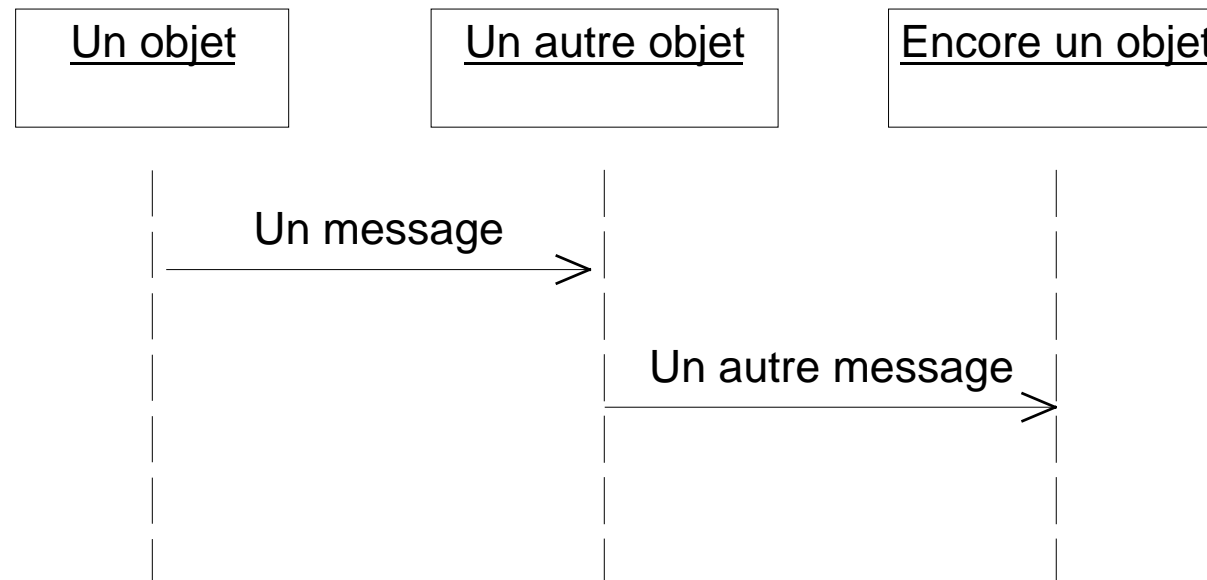
Diagrammes de collaboration

- Représentation des patterns



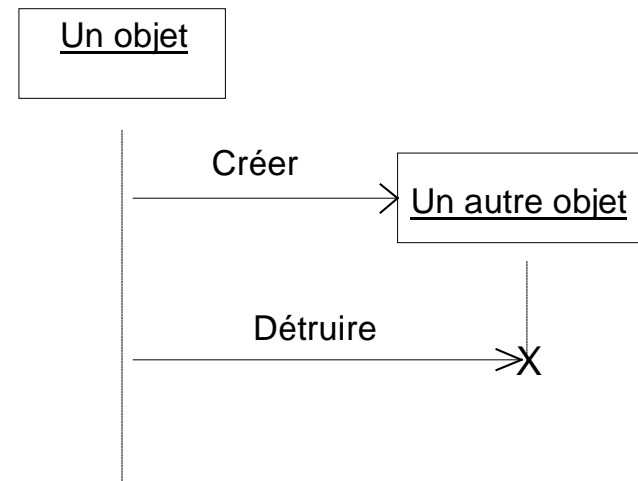
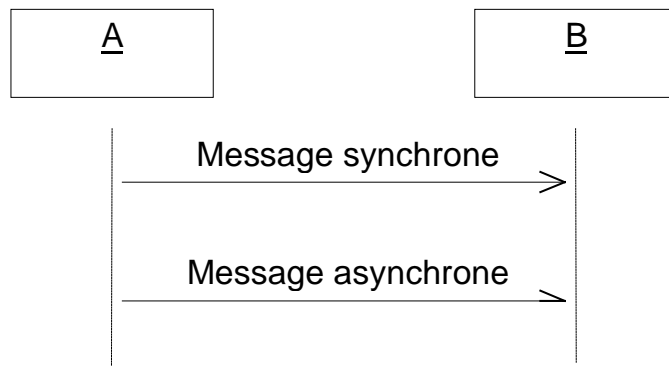
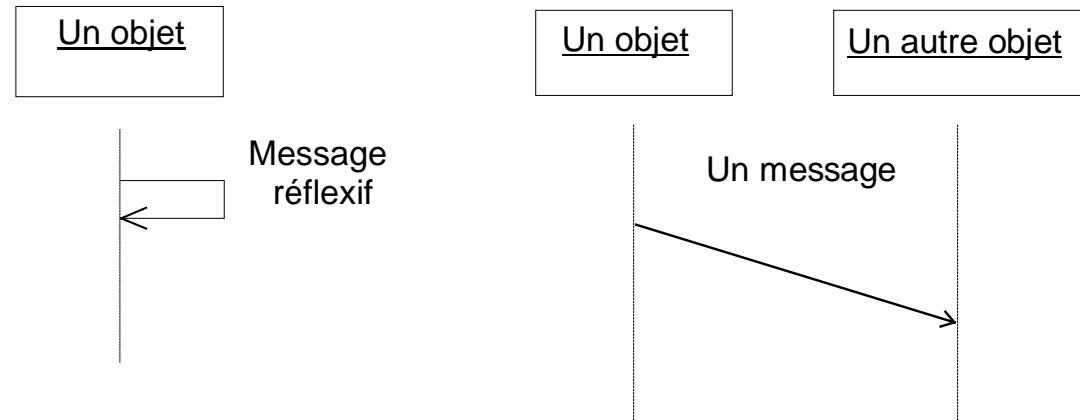
Diagrammes de séquence

- Représentation temporelle d'une interaction



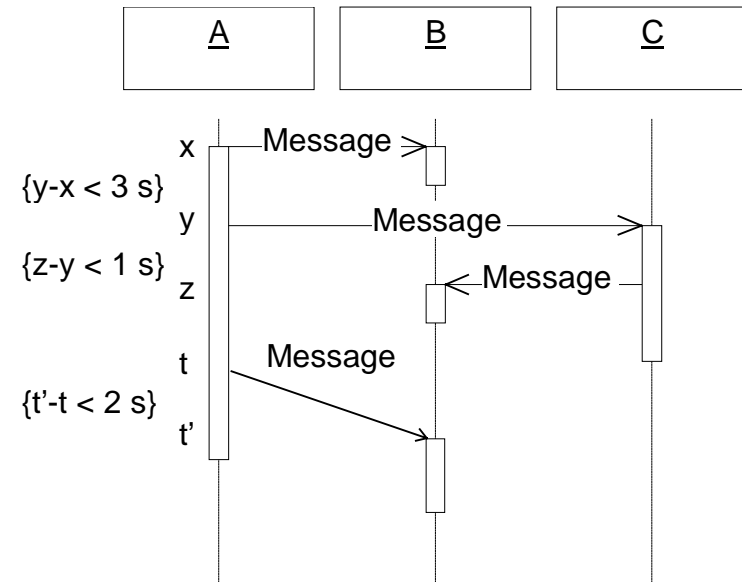
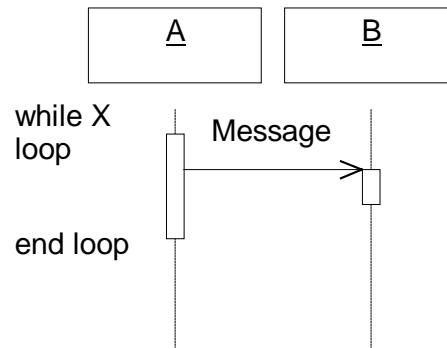
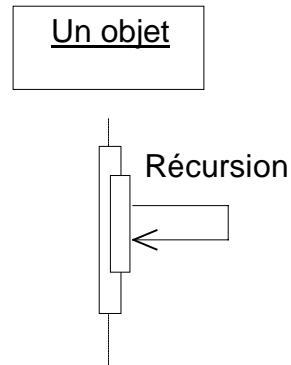
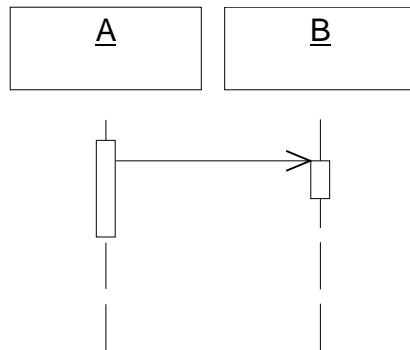
Diagrammes de séquence

- Exemples



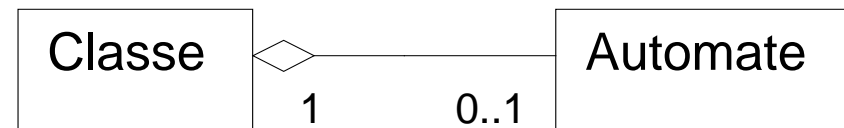
Diagrammes de séquence

- Exemples



Diagrammes d'états-transitions

- Représentation des automates
 - *Statecharts* (David Harel)



●
Etat initial

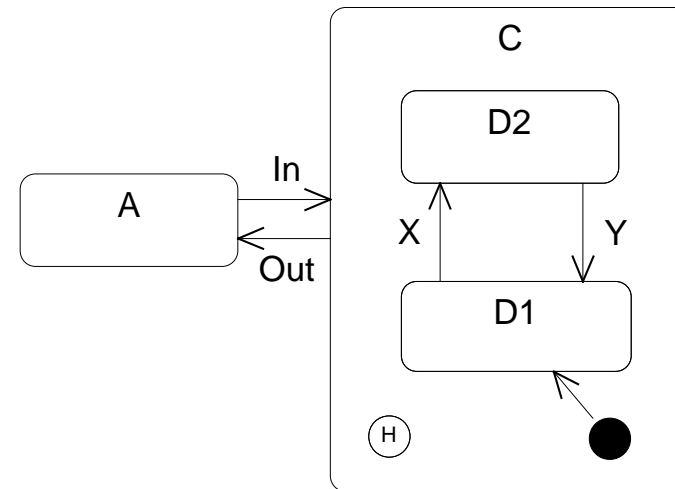
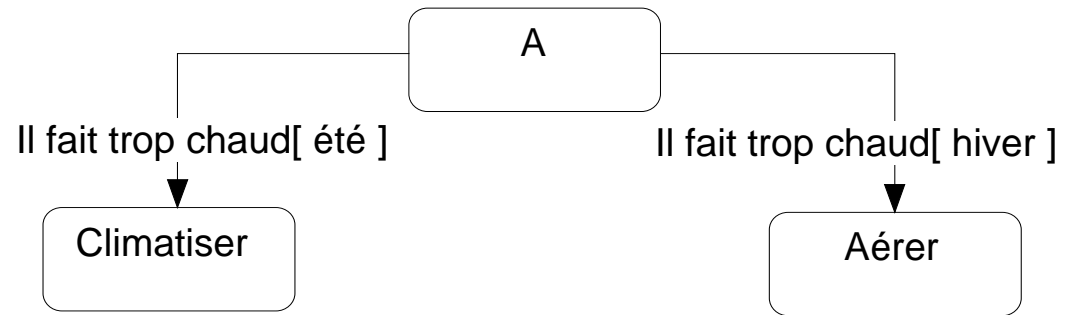
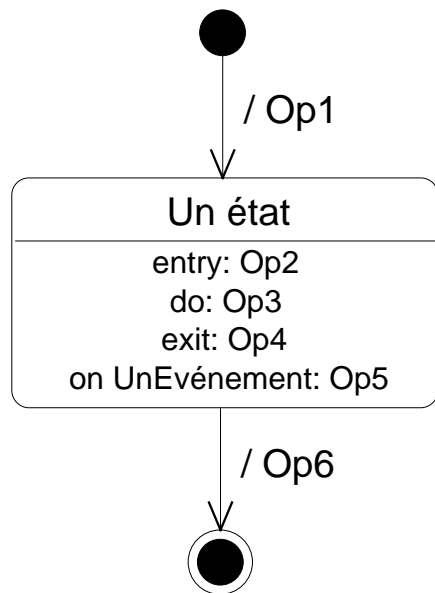


●
Etat final



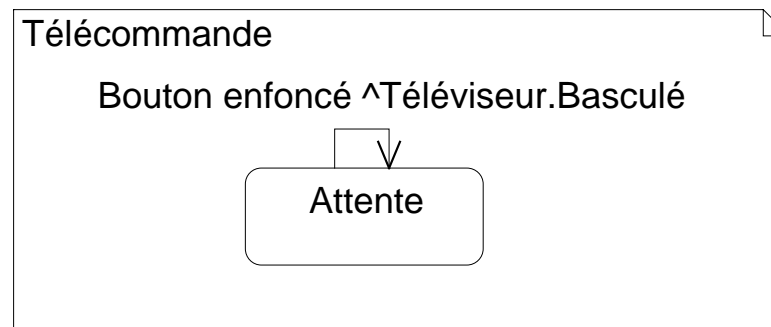
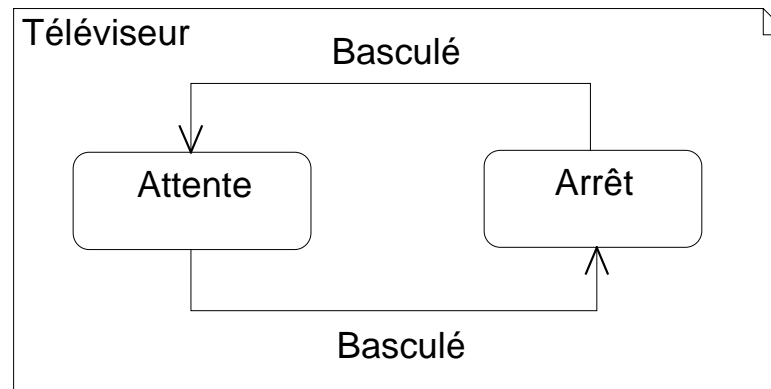
Diagrammes d'états-transitions

- Exemples



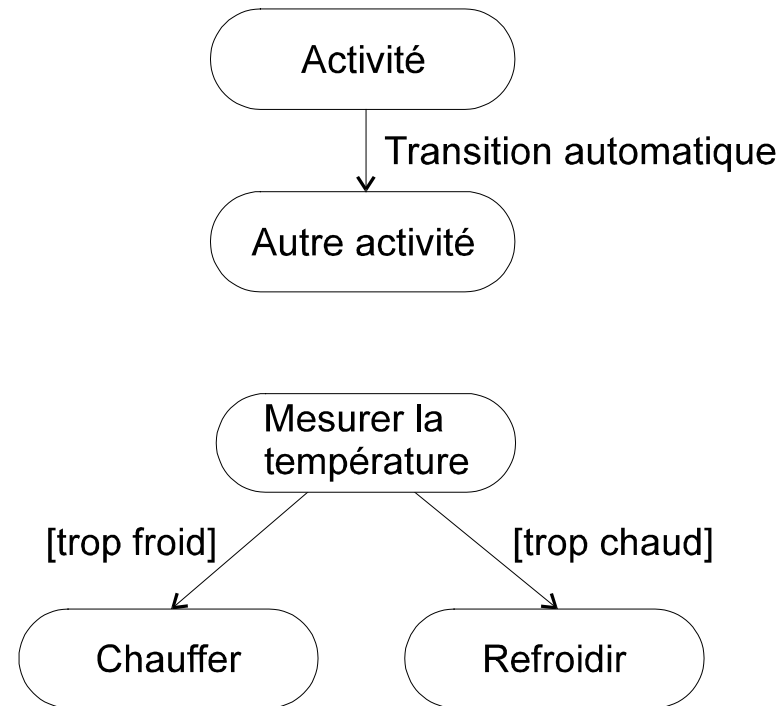
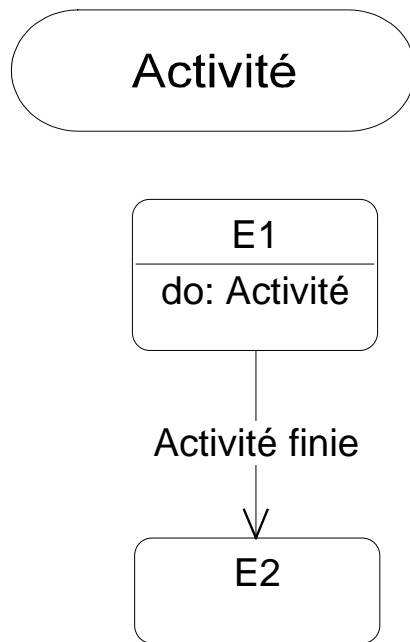
Diagrammes d'états-transitions

- Exemples



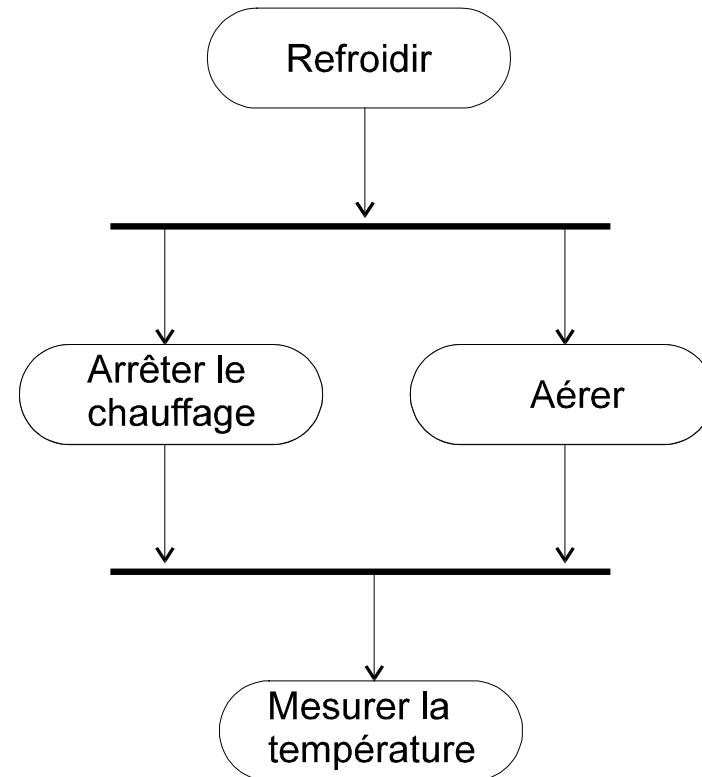
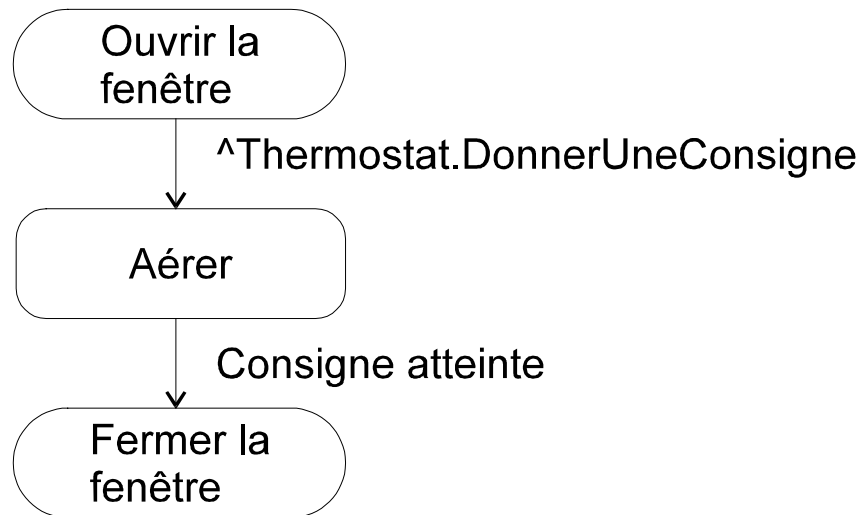
Diagrammes d'activités

- Représentation d'un automate du point de vue des activités



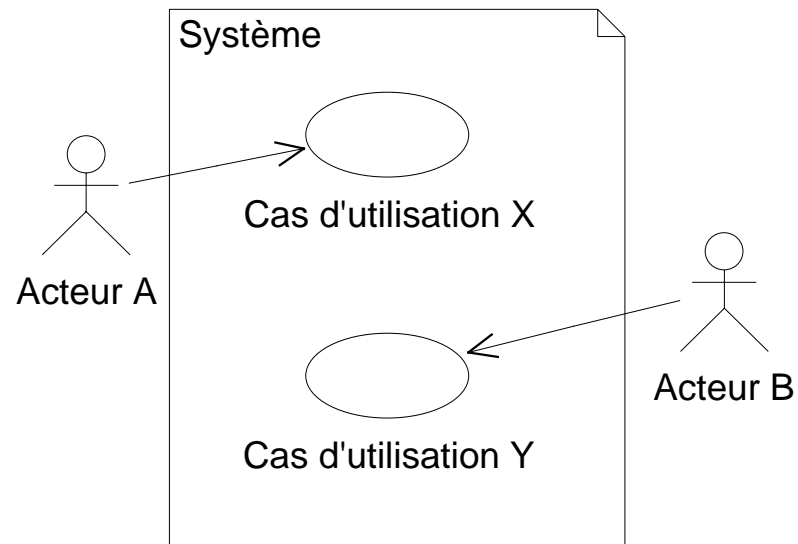
Diagrammes d'activités

- Exemples



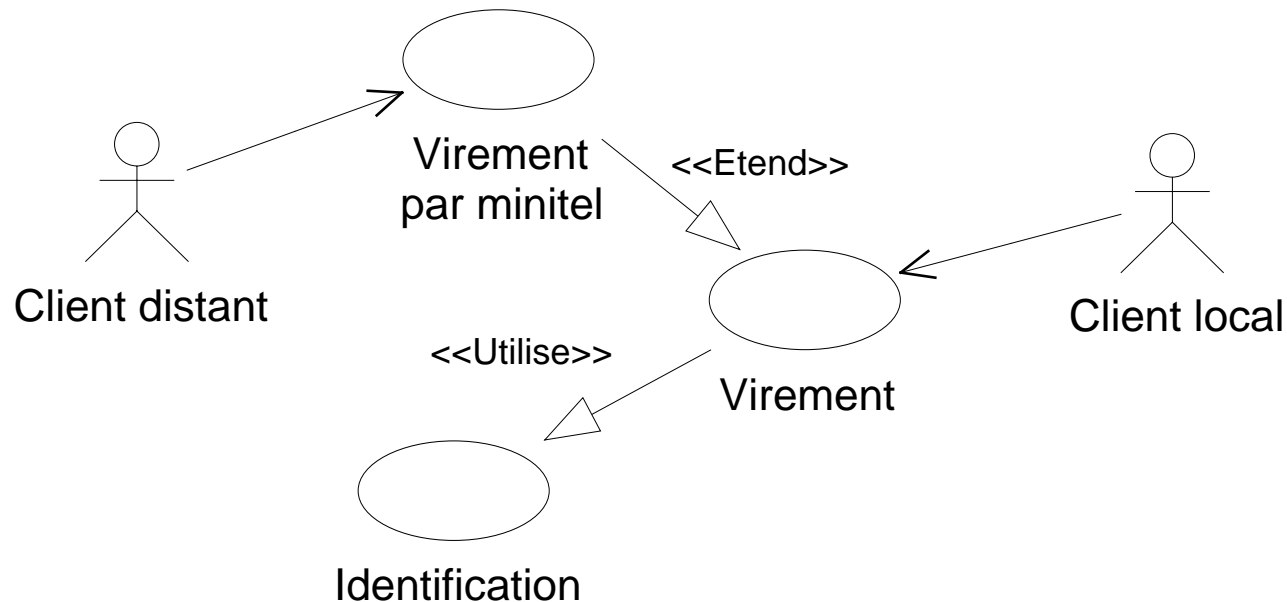
Les diagrammes de cas d'utilisation

- Formalisés par I. Jacobson (Use Case)



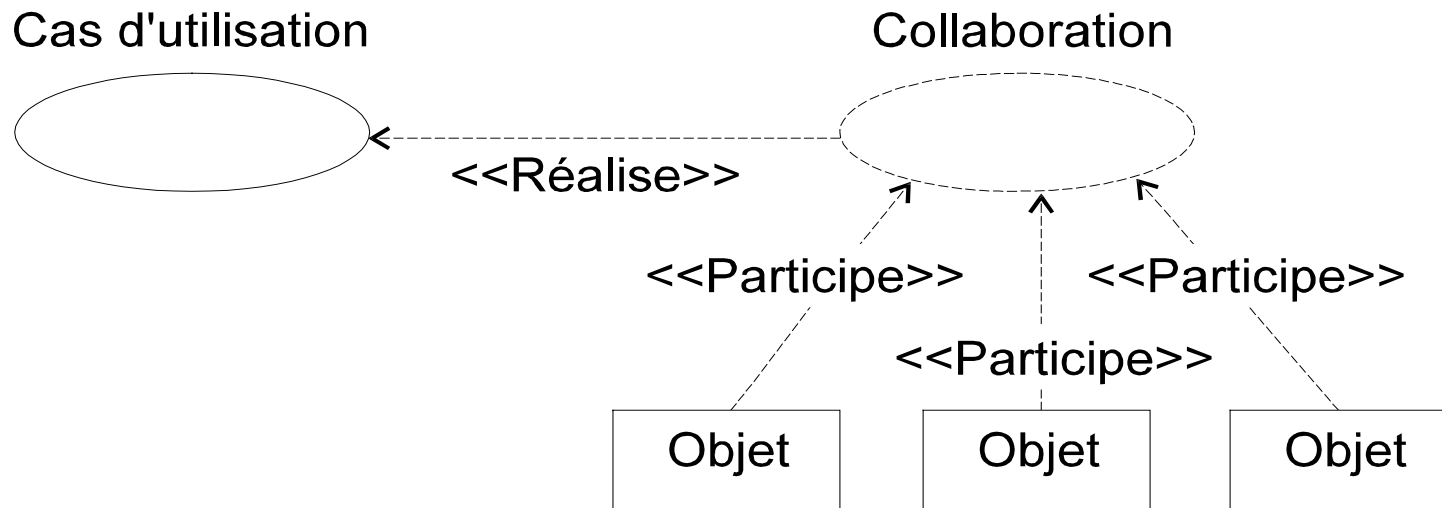
Les diagrammes de cas d'utilisation

- Relations entre cas d'utilisations



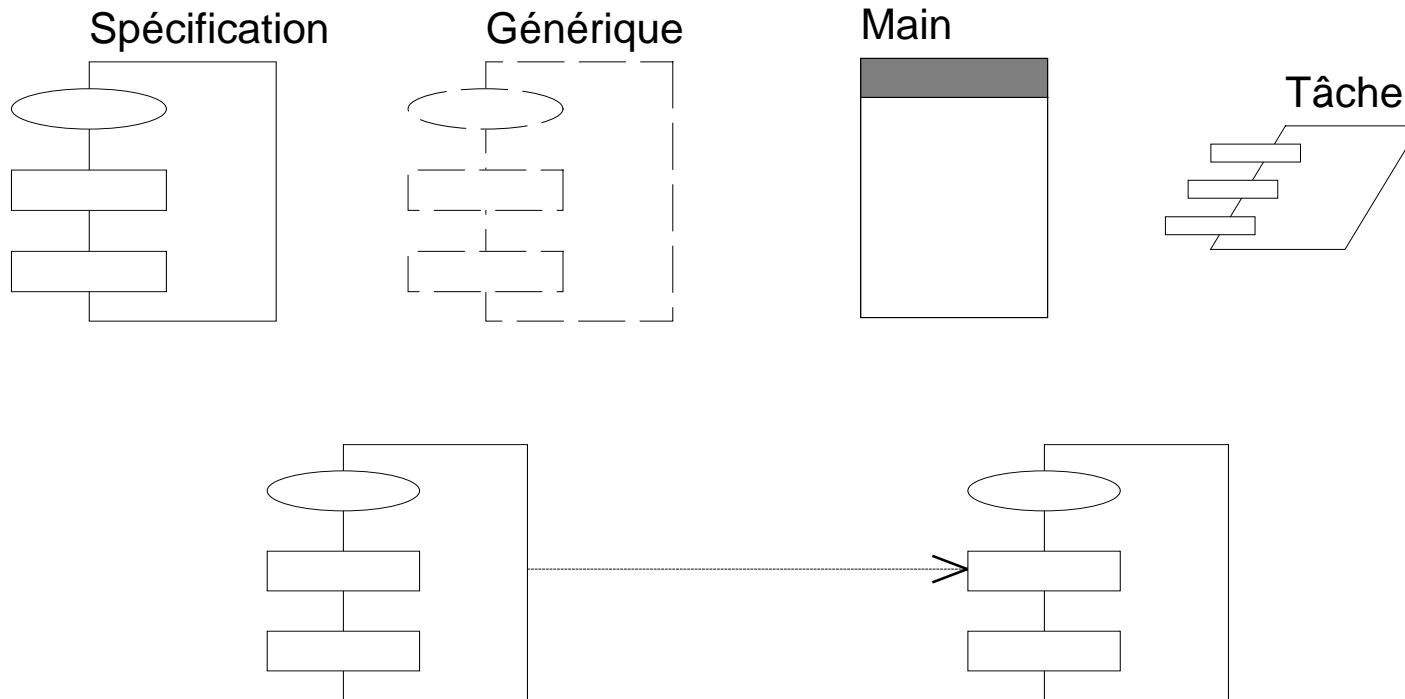
Les diagrammes de cas d'utilisation

- Transition vers les objets

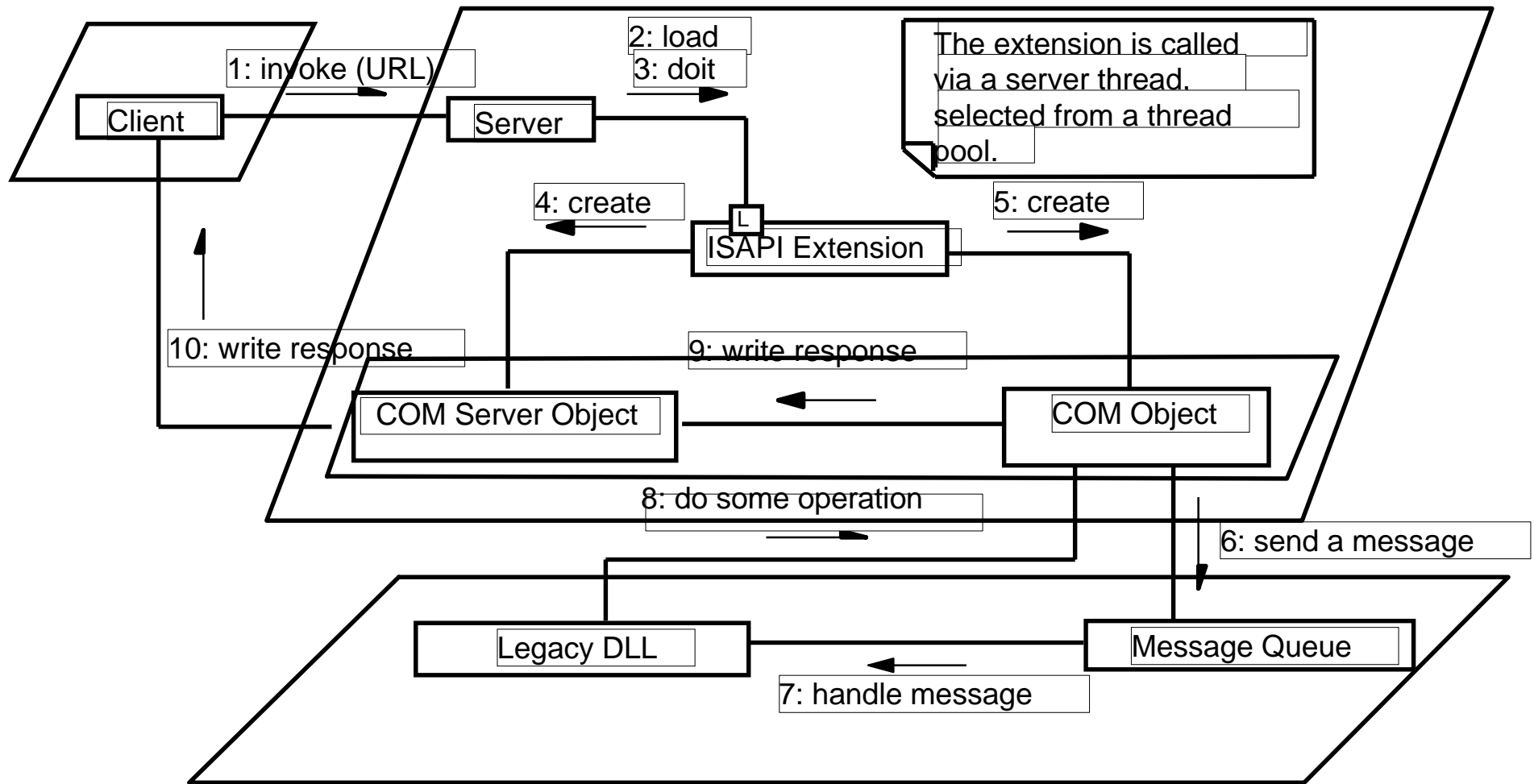


Diagrammes de composants

- Représentation des éléments de réalisation

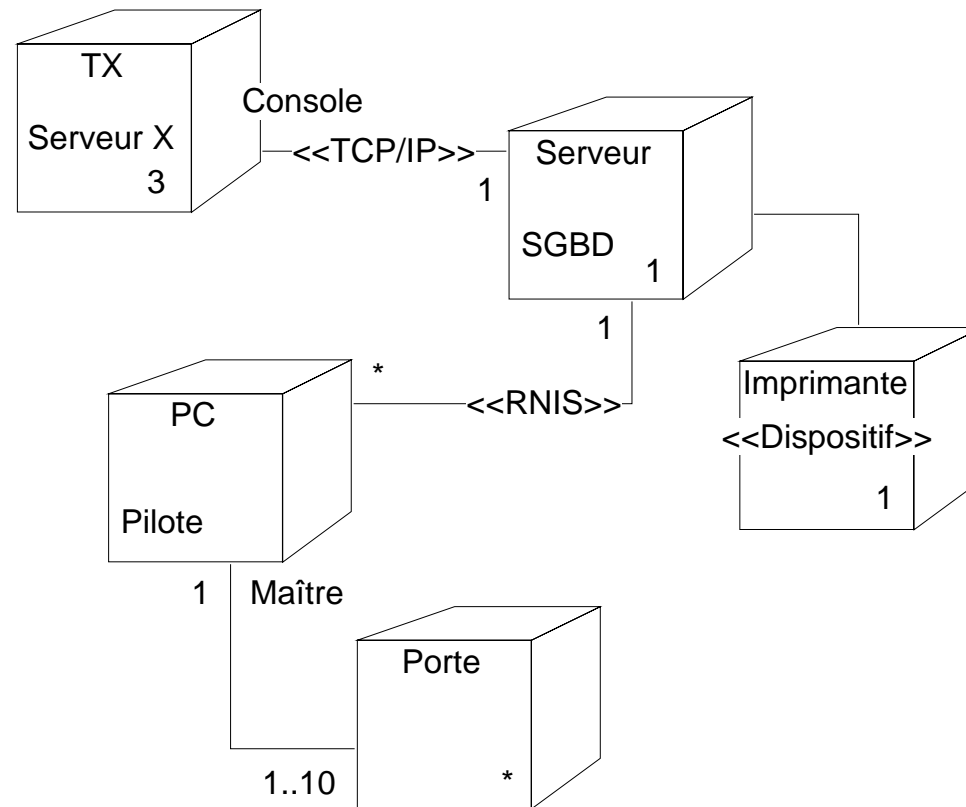


Diagrammes de composants

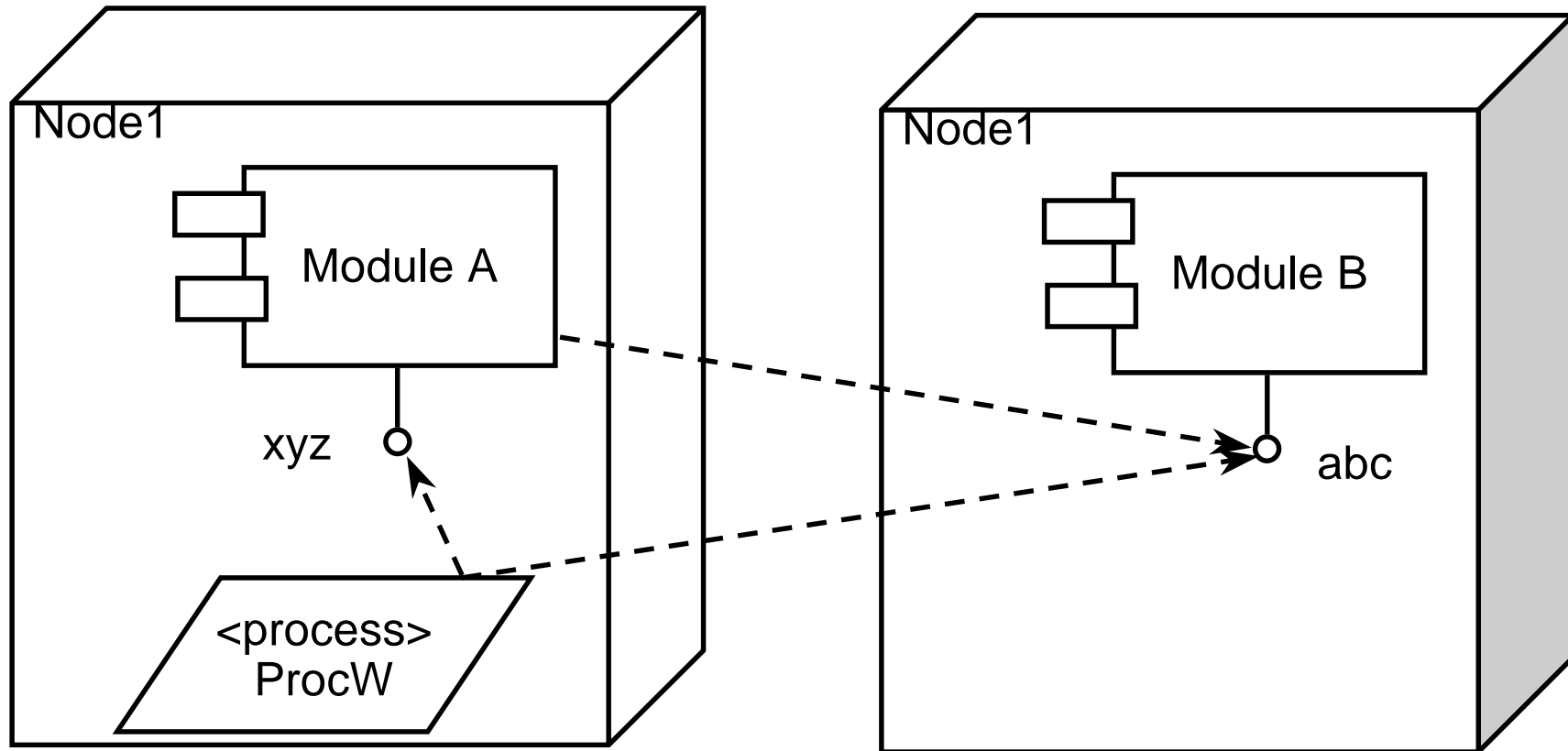


Diagrammes de déploiement

- Architecture matérielle et répartition du logiciel



Diagrammes de déploiement





Vers un processus unifié

Objectifs

- Construire des modèles de systèmes
- Organiser le travail
- Gérer le cycle de vie de A à Z
- Gérer le risque
- Obtenir de manière répétitive des produits de qualité constante

Caractéristiques du processus

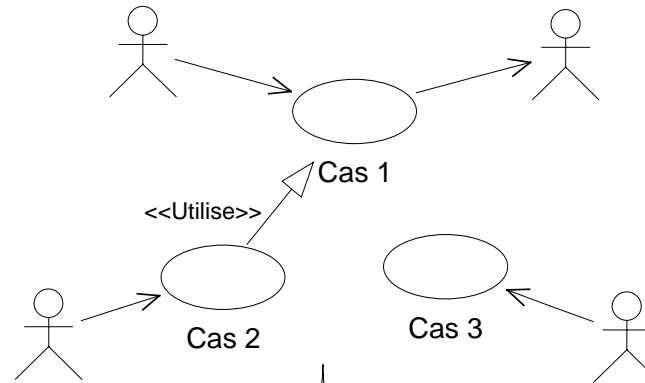
- Dirigé par les cas d'utilisation
- Centré sur l'architecture
- Itératif
- Incrémental

Dirigé par les cas d'utilisation

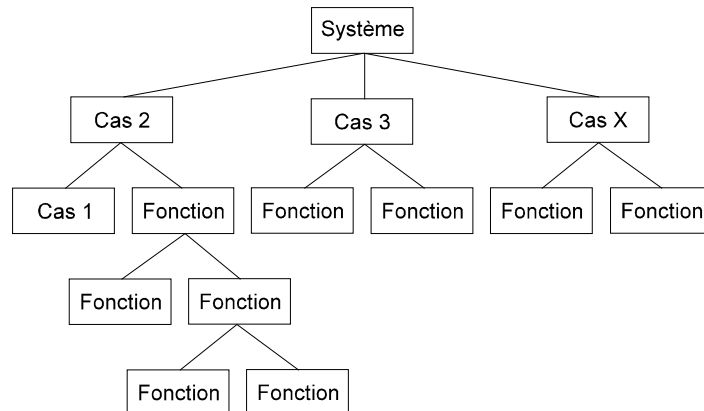
- Fil conducteur de toutes les activités



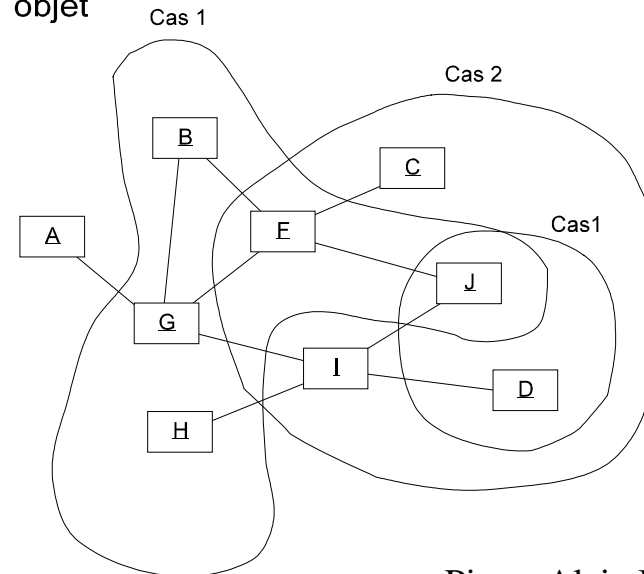
Dirigé par les cas d'utilisation



Décomposition structurée



Décomposition objet

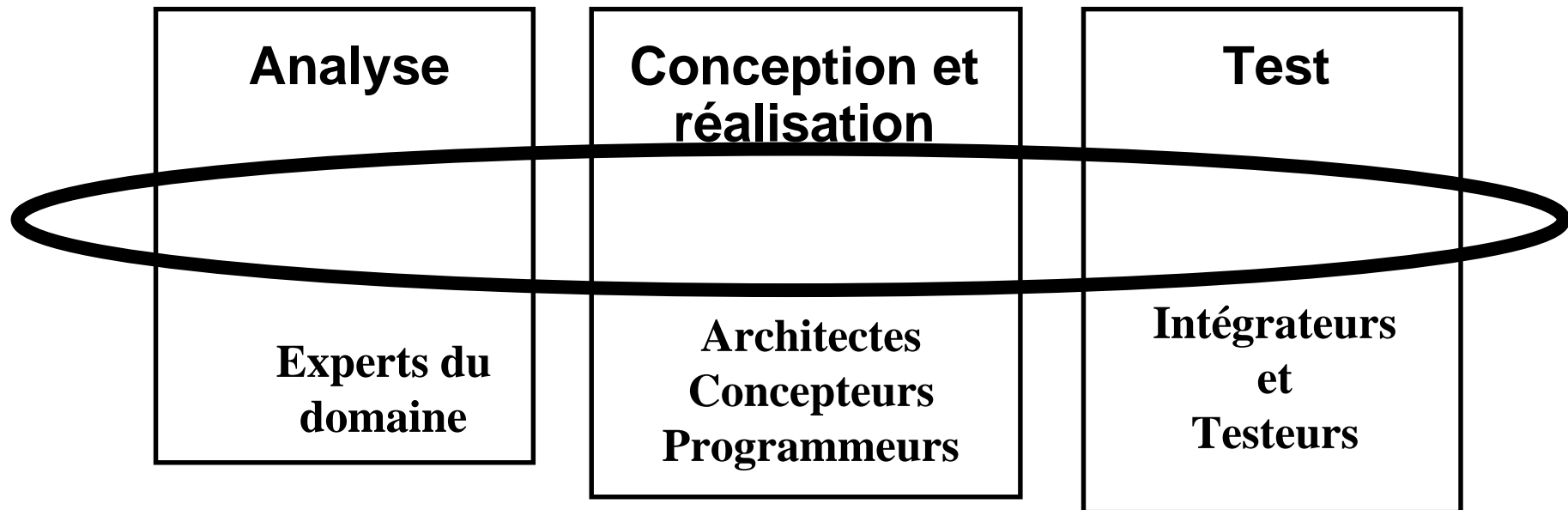


Les cas d'utilisation et les tests

- En Analyse
 - Modélisation en cas d'utilisation
 - Définition des cas de test
- En conception
 - Génération des cas de test depuis les diagrammes d'interaction et les automates d'états finis

Organisation du travail

- Découpage par cas d'utilisation

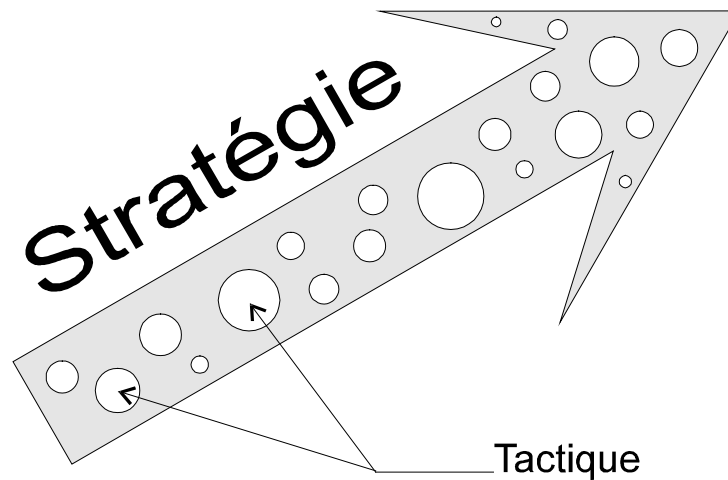


Centrage sur l'architecture

- Recherche de la forme générale du système dès le début
- Approche systématique pour trouver une “bonne” architecture
 - Support des cas d'utilisation
 - Adaptation aux changements
 - Pour et avec la réutilisation
 - Compréhensible intuitivement

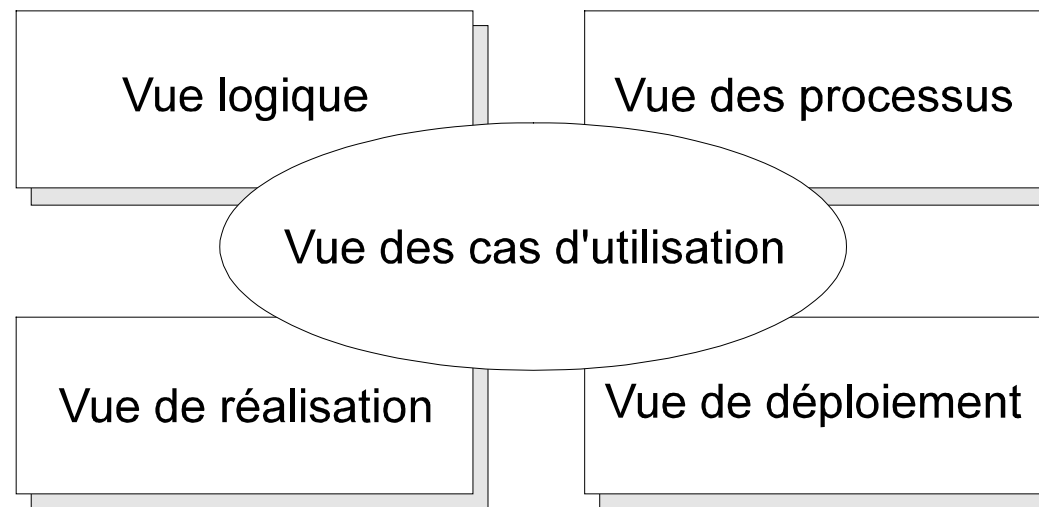
Architecture logicielle

- Architecture =
Eléments + Formes + Motivations
- Architecture = Stratégie + Tactique



La vision de l'architecte

- Il n'existe pas une seule manière de regarder un système
 - Philippe Kruchten, le modèle 4 + 1 vues, IEEE Software, Nov. 95



Le modèle 4 + 1 vues

- La vue logique
- La vue de réalisation
- La vue des processus
- La vue de déploiement
- La vue des cas d'utilisation

La vue logique

- Aspects statiques et dynamiques
- Les éléments
 - Les objets
 - Les classes
 - Les collaborations
 - Les interactions
 - Les paquetages <<Catégorie>>

La vue de réalisation

- Organisation des modules dans l'environnement de développement
- Les éléments
 - Les modules
 - Les sous-programmes
 - Les tâches (en tant qu'unités de programme, comme en Ada)
 - Les paquetages << sous-système >>

La vue des processus

- Décomposition en flots d'exécution et synchronisation entre ces flots
- Les éléments
 - Les tâches
 - Les threads
 - Les processus
 - Les interactions

La vue de déploiement

- Les ressources matérielles et l'implantation du logiciel dans ces ressources
- Les éléments
 - Les noeuds
 - Les modules
 - Les programmes principaux

La vue des cas d'utilisation

- La colle entre les autres vues
- Les éléments
 - Les acteurs
 - Les cas d'utilisation
 - Les classes
 - Les collaborations

Récapitulatif

	Vue des cas d'utilisation	Vue logique	Vue de réalisation	Vue des processus	Vue de déploiement
Diagramme de cas d'utilisation	Acteurs Cas d'utilisation				
Diagramme de classes		Classes Relations			
Diagramme d'objets	Objets Liens	Classes Objets Liens			
Diagramme de séquence	Acteurs Objets Messages	Acteurs Objets Messages		Objets Messages	
Diagramme de collaboration	Acteurs Objets Liens Message	Acteurs Objets Liens Messages		Objets Liens Messages	

Récapitulatif (suite)

	Vue des cas d'utilisation	Vue logique	Vue de réalisation	Vue des processus	Vue de déploiement
Diagramme d'états- transitions	Etats Transitions	Etats Transitions		Etats Transitions	
Diagramme d'activité	Activités Transitions	Activités Transitions		Activités Transitions	
Diagramme de composants			Composants	Composants	Composants
Diagramme de déploiement					Noeuds Liens

Architecture, processus et organisation

- Les processus et l'organisation doivent être adaptés à l'architecture
 - Un processus pour l'architecture générale
 - Un processus par application, composant système ou couche
 - Un processus par type de système

Une bonne architecture facilite

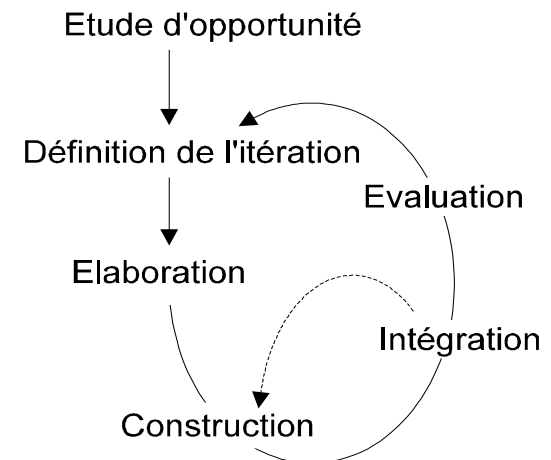
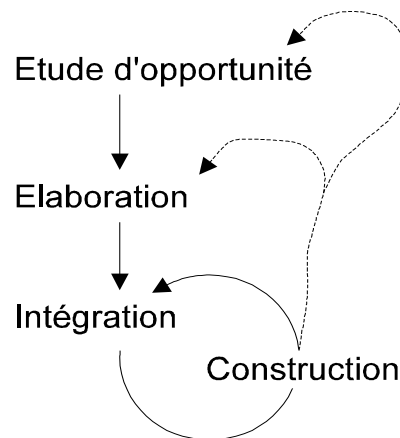
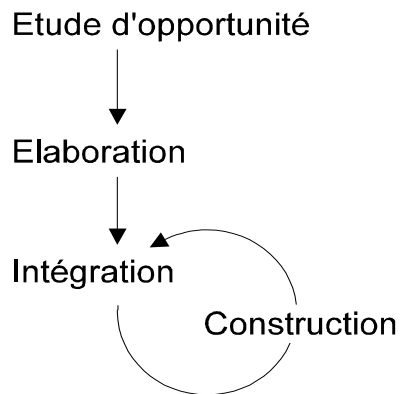
- L'assemblage des composants pour des besoins génériques
- Le partage de composants réutilisables
- La navigation depuis les besoins jusqu'au code et réciproquement
- La responsabilisation des développeurs
- L'adaptation et l'évolution

Approche itérative et incrémentale

- Segmentation du travail
- Concentration sur les besoins et les risques
- Les premières itérations sont des prototypes
 - Expérimentation et validation des technologies
 - Planification
- Les prototypes définissent le noyau de l'architecture

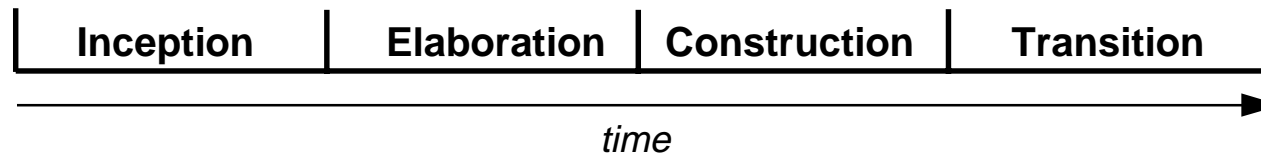
Approche itérative et incrémentale

- L'ordonnancement des itérations est basée sur les priorités entre cas d'utilisation et sur l'étude du risque



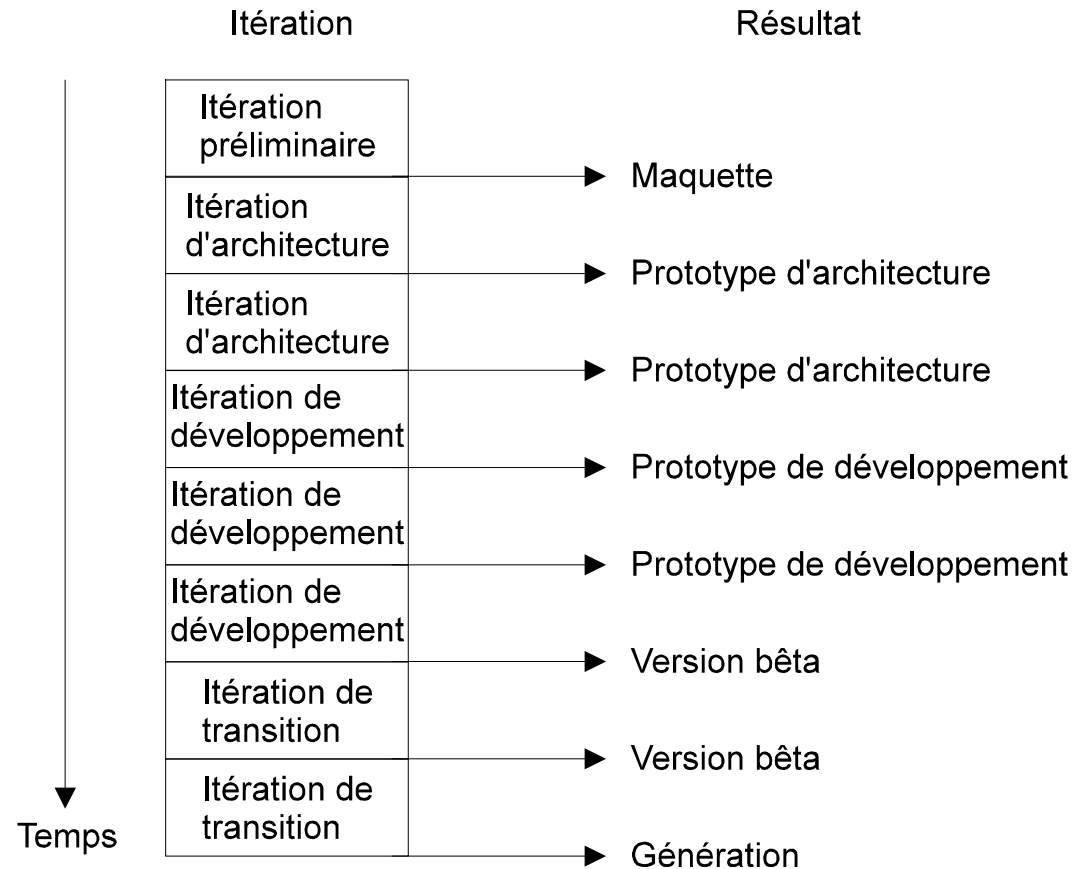
Vue de l'encadrement

- Des phases
 - Inception (étude d'oportunité)
 - Elaboration (architecture, planning)
 - Construction
 - Transition



Vue technique

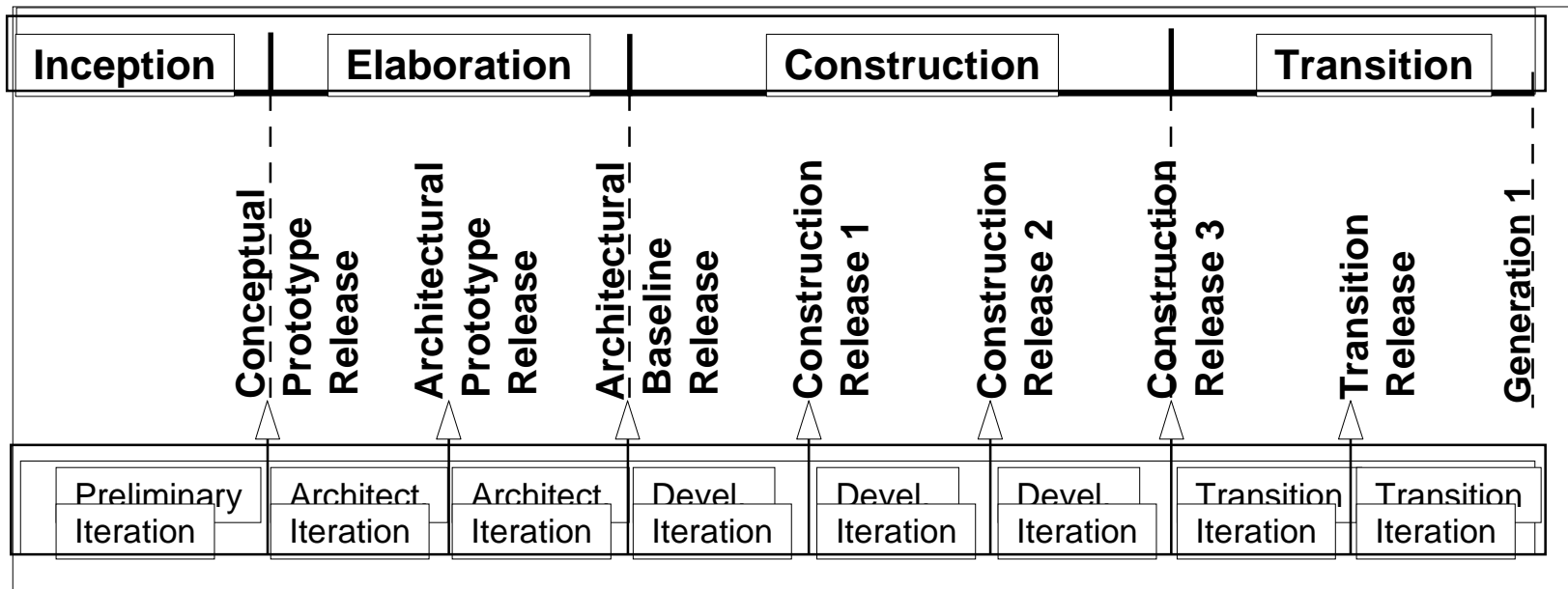
- Des itérations



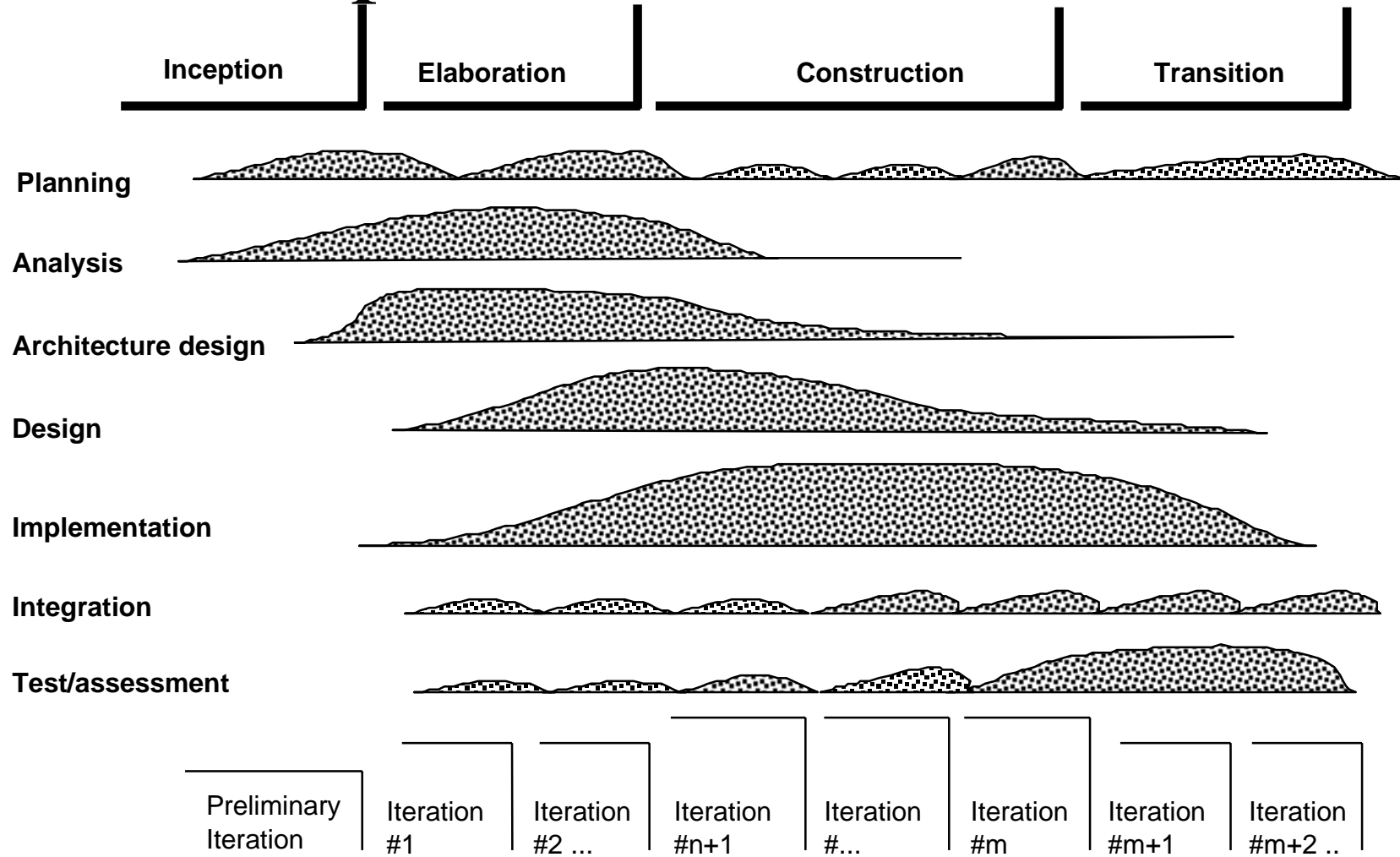
Synchronisation des deux vues

- Itérations
 - Chaque cycle donne une génération
 - Chaque cycle est décomposé en phases
 - Chaque phase comprend des itérations
- Incréments
 - Le logiciel évolue par incrément
 - Une itération correspond à un incrément
 - Les itérations peuvent évoluer en parallèle

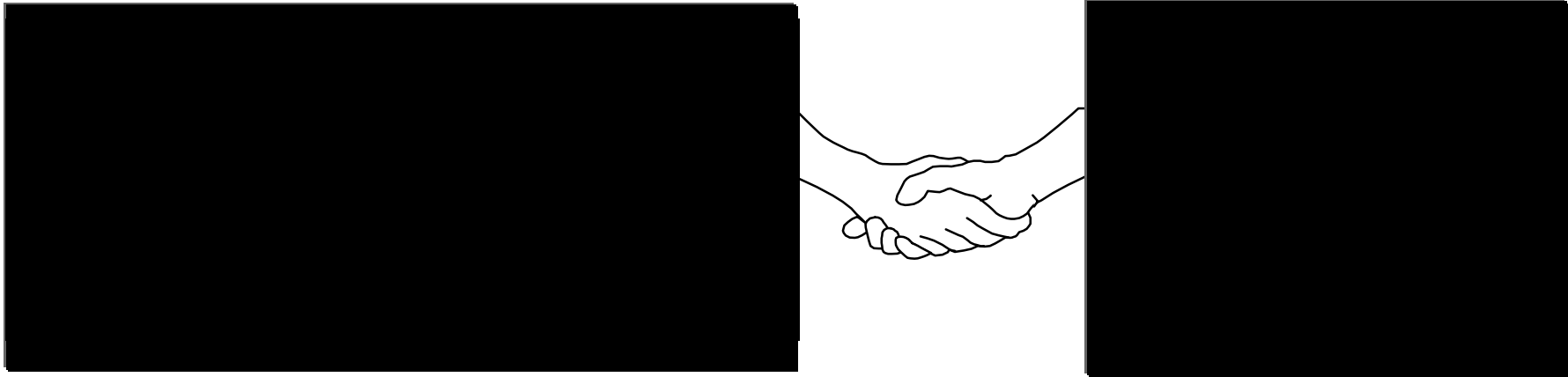
Synchronisation des deux vues



Répartition des efforts



UML en résumé



- **Convergence aujourd'hui**
- **L'unification conduit à la standardisation**

- **Convergence dans le futur**
- **Consensus autour d'un cadre directeur**



La suite de l'histoire

Prochaines étapes

- 1^{er} semestre 97
 - Consolidation de la proposition UML 1.0
- Septembre 97
 - Approbation du standard par le comité technique de l'OMG

Pour en savoir plus

- www.rational.com
- otug@rational.com
- www.essaim.univ-mulhouse.fr
- uml@essaim.univ-mulhouse.fr
 - inscription automatique par mail à
 - majordomo@essaim.univ-mulhouse.fr
 - avec dans le corps du message : `subscribe uml`

Pour en savoir plus

- Modélisation objet avec UML
 - *Pierre-Alain Muller*, Eyrolles, 430 pages
- Sommaire
 - Genèse d'UML
 - Approche objet
 - Notation UML
 - Encadrement des projets objet
 - Etude de cas