UML – Introduction

Notion 'système d'information'

Diagrammes Uml

Contenu

I. Introduction aux méthodologies des systèmes d'information

- Le système d'information
- le cycle de vie d'un système d'information

II. L'approche UML

- les concepts objet
- Modéliser avec UML

I. Le système d'information

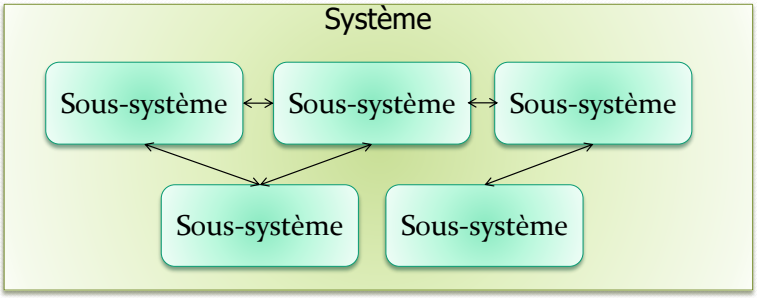
a. Définition du terme 'Système'

Un système est un **ensemble d'éléments** en **interaction** dans lequel **chacun concourt** à **l'objectif commun** ou à la **finalité.**

Le **comportement** de **chaque élément** a un **sens** par rapport à **l'objectif** (la finalité) **global**.

I. Le système d'information

a. Définition du terme 'Système'



Un système peut être décomposé en plusieurs sous-systèmes en interaction

I. Le système d'information

b. Les systèmes dans une organisation

Carina Roels 5

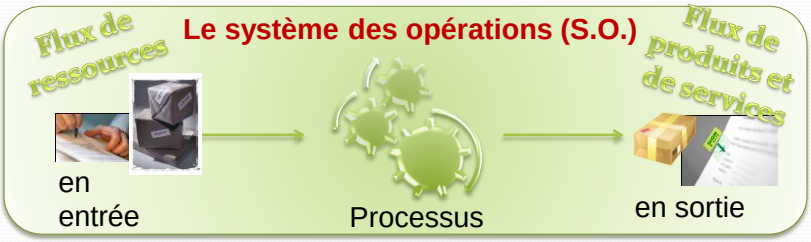
Le système des opérations transforme les flux de ressources (entrée) en flux de produits et de services (sortie) par l'intermédiaire des processus définis.

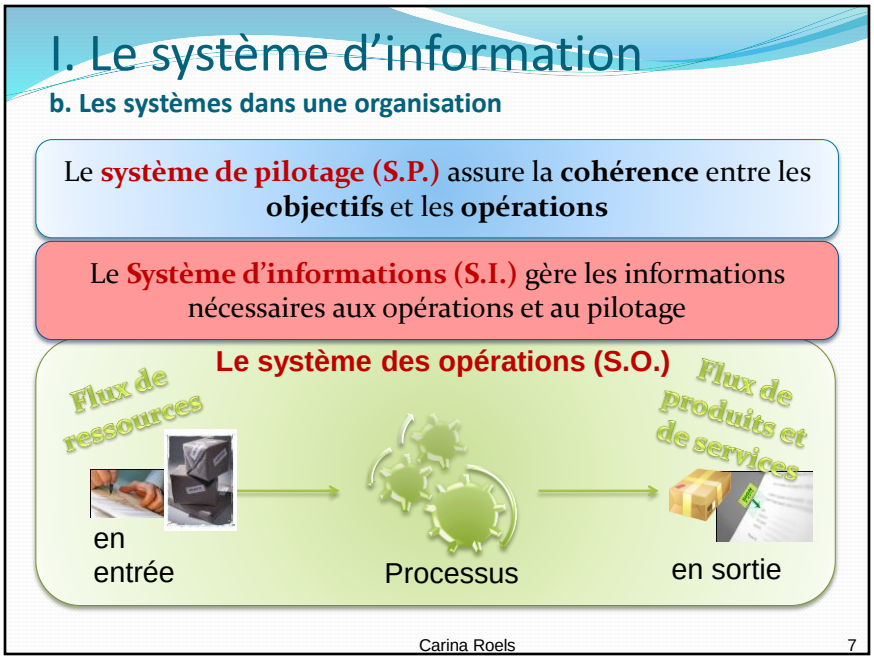
I. Le système d'information

b. Les systèmes dans une organisation

Le système de pilotage (S.P.)

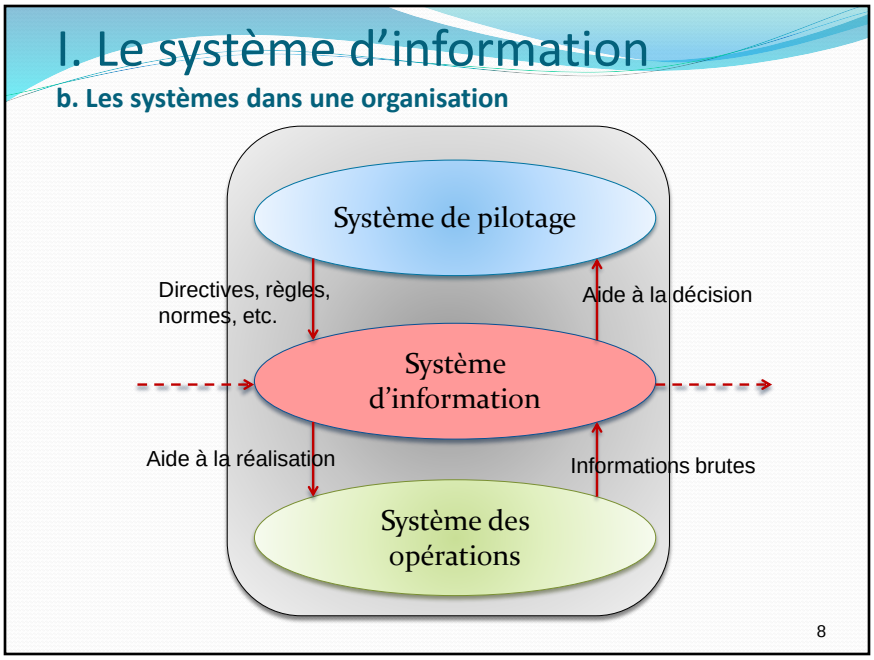
Le **système de pilotage (S.P.)** assure la **cohérence** entre les **objectifs** et les **opérations**





La gestion des informations est nécessaire pour :

- Déterminer et contrôler les objectifs
- Coordonner les opérations (cohérence globale sur une période donnée)
- Suivre les opérations
- Evaluer les performances

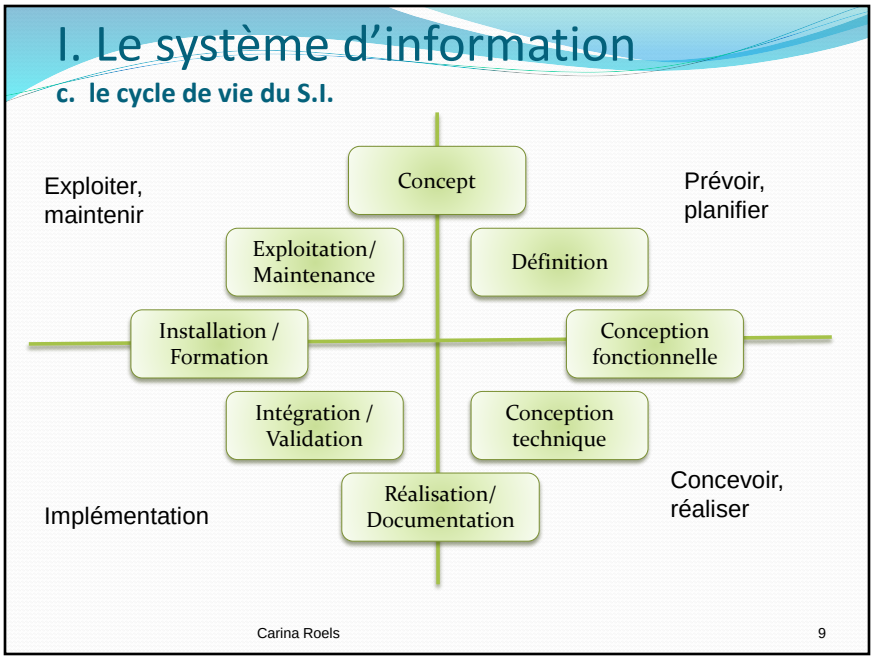


Le système d'informations est le lien entre le système de pilotage et le système des opérations.
I sert a acquérir, à traiter et à stocker des informations :

- Venant du système de pilotage et servant au système des opérations pour des actions courantes.
- Venant du système des opérations et servant au système de pilotage pour le prise de décisions.

A cela doivent également s'ajouter des informations :

- Venant de l'extérieur et étant nécessaires pour le système des opérations (ex. des commandes, des livraisons fournisseur) et pour le système de pilotage ex. des lois, règlementations, etc.)
- Transmises à l'extérieur (Ex. des factures, des échanges avec d'autres organisations, etc.)



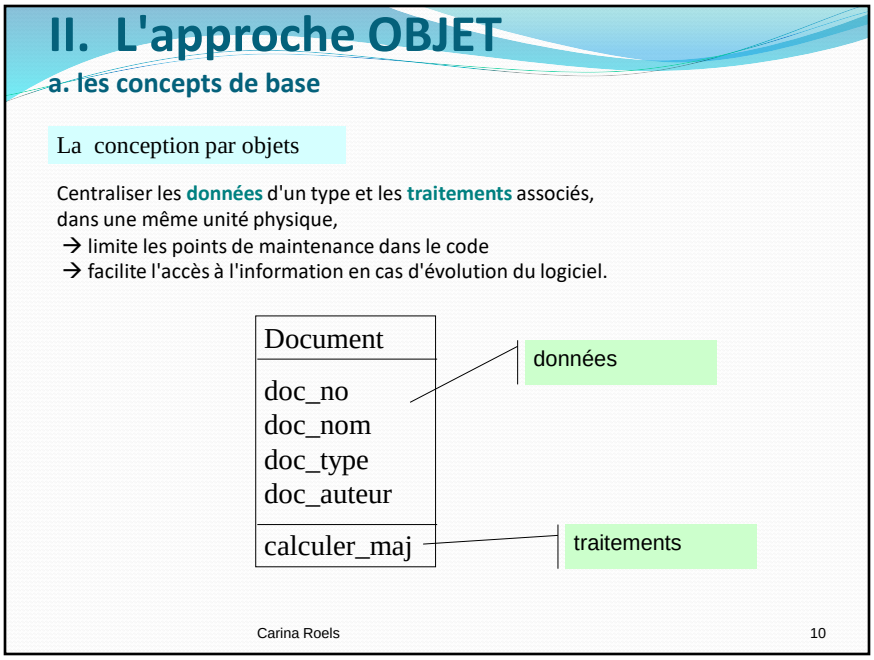
La D.S.I. est en charge de la totalité du cycle de vie du S.I. et de ses sous-systèmes.
Le cycle de vie du S.I. peut être décomposé en 4 grandes phases :

La phase **de prévision et de planification** (appelé parfois 'genèse')
Formaliser l'idée (le concept) du projet et définir le système. L'idée est alors transformé en projet, pour lequel une Note de cadrage (ou Brief projet) doit être rédigée. Une première planification (macro-planning) permettra de définir les grandes lignes du projet.

La phase **de conception et de réalisation**
La conception fonctionnelle est réalisé à l'aide de modèles de données et de processus.
La conception technique sert à introduire des contraintes techniques dans la conception (types de matériel, systèmes d'exploitation, nature des réseaux, types de SGBD, etc.)
la réalisation (programmation, tests et validations intermédiaires) est basé sur les dossiers de conception. Une documentation associée est également réalisée.

La phase **d'Implémentation**
Deux types d'intégration peuvent être réalisées :
- l'intégration des composants matérielles et logicielles
- l'intégration dans la structure existante de l'organisation
La solution nécessite une validation par les directions/fonctions concernés.
Suite à la validation, la solution peut être déployée sur les serveurs et postes de travail de l'organisation
Une formation et accompagnement des utilisateurs leur permet de s'approprier le nouveau système.
En cas de basculement d'un ancien vers un nouveau système, le lancement comprend également tous les travaux préparatoires.

La phase **d'exploitation et de maintenance**
Exploiter : utiliser le système pour les tâches de production et de support
Maintenance : garantir le niveau opérationnel



Une structure de données (ici : document) est manipulée par des fonctions (ici : la fonction calculer_maj).
L ’ensemble des propriétés cohérentes et les traitements associés s ’appelle un **objet**.

Un objet possède :

- une identité (un nom)
- des attributs qui caractérisent l ’état de l ’objet
- un ensemble d ’opérations (méthodes) qui définissent le comportement

Un **objet** est une instance de classe (une occurrence d ’un type abstrait)

Une **classe** est un type de données abstrait, caractérisé par des propriétés (attributs et méthodes) communes à des objets.

II. L'approche OBJET

a. les concepts de base

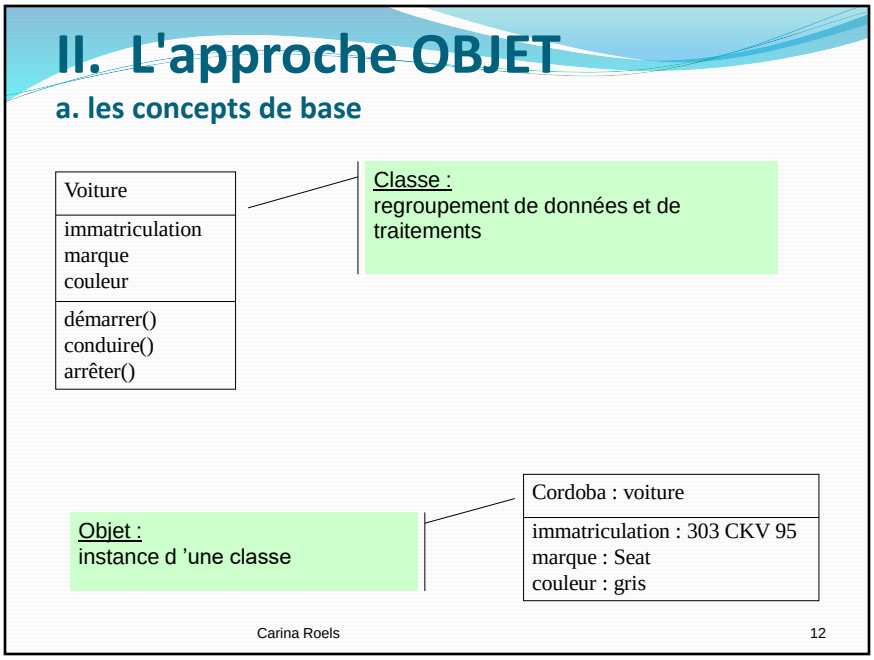
OBJET

n'importe quoi d'identifiable

- entité physique (ex. une voiture)
- entité intangible (ex. une maladie, un service…)

LA CONCEPTION PAR LES OBJETS

Démarche qui conduit à des architectures logicielles **fondées sur les objets** que tout système ou sous-système manipule, plutôt que sur " la " fonction qu'il est censé réaliser.

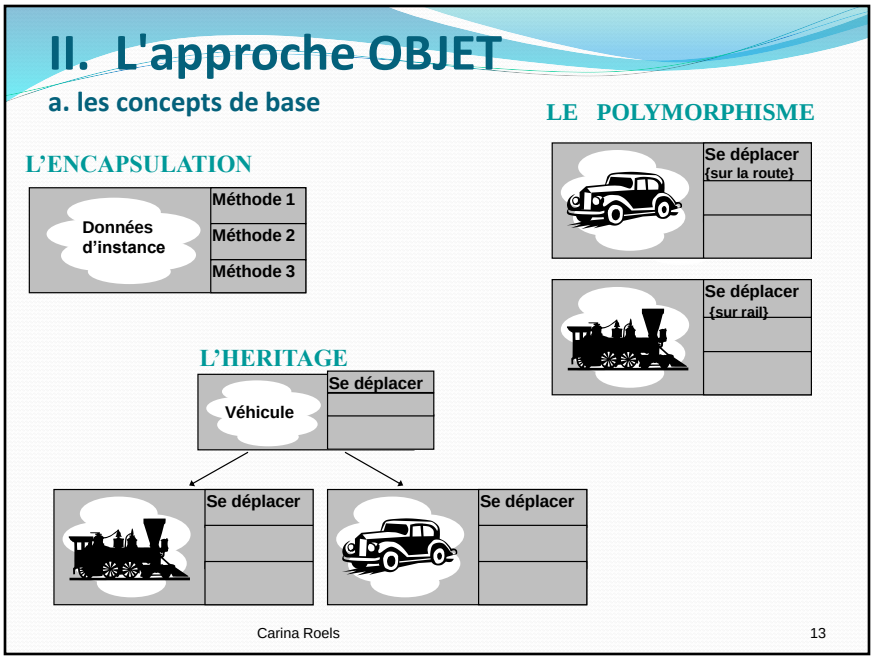


CLASSE : terme générique, pour décrire des ensembles de structures de données.

OBJET : instance d'une classe.

Les objets ont 3 propriétés qui les rendent particulièrement utiles (les piliers pour créer, assembler, réutiliser les objets)

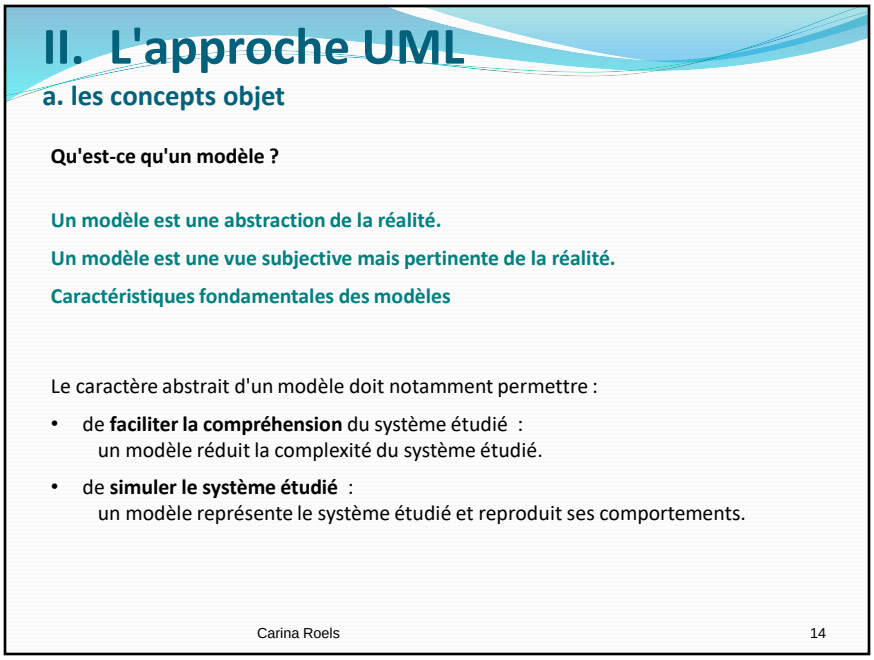
- l'encapsulation
- l'héritage (généralisation / spécialisation)
- le polymorphisme



- **L'encapsulation**
  - Consiste à masquer les détails d'implémentation d'un objet, en définissant une interface.
  - L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet.
  - L'encapsulation facilite l'évolution d'une application car elle stabilise l'utilisation des objets : on peut modifier l'implémentation des attributs d'un objet sans modifier son interface.
  - L'encapsulation garantit l'intégrité des données, car elle permet d'interdire l'accès direct aux attributs des objets (utilisation d'accesseurs).

- **L'héritage**
  - mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.
  - Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines. Plusieurs classes peuvent être généralisées en une classe qui les factorise, afin de regrouper les caractéristiques communes d'un ensemble de classes. La spécialisation et la généralisation permettent de construire des hiérarchies de classes. L'héritage peut être simple ou multiple.
  - L'héritage évite la duplication et encourage la réutilisation.

- **Le polymorphisme** représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes.
  - Le polymorphisme augmente la généricité du code.



Un modèle est une abstraction de la réalité
L'abstraction est un des piliers de l'approche objet.

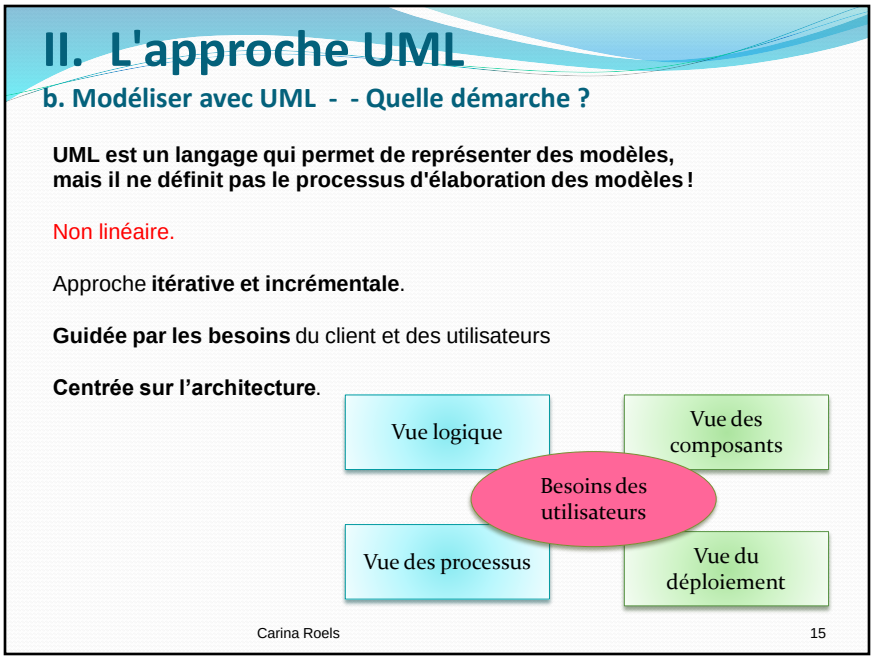
•Il s'agit d'un processus qui consiste à identifier les caractéristiques intéressantes d'une entité, en vue d'une utilisation précise.

•L'abstraction désigne aussi le résultat de ce processus, c'est-à-dire l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur.

Un modèle est une vue subjective mais pertinente de la réalité

•Un modèle définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue très subjective de la réalité.

•Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente.



Une démarche itérative et incrémentale

La modélisation d'un système complexe se fait en plusieurs fois, en affinant la représentation (compréhension) par étapes.

Le but est de mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes.

Une démarche guidée par les besoins du client et des utilisateurs

Le périmètre du système à modéliser est défini par les besoins des utilisateurs.

Les besoins des utilisateurs servent de fil rouge, tout au long du cycle de développement (itératif et incrémental) :

- Lors de la phase d'analyse → clarification, affinage et validation des besoins des utilisateurs.
- Lors de la phase de conception et de réalisation → vérification de la prise en compte des besoins des utilisateurs.
- Lors de la phase de test → vérification de la satisfaction des besoins.

Une démarche centrée sur l'architecture

Une architecture adaptée est la pierre angulaire d'un développement.

Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).

II. L'approche objet

b. Modéliser avec UML - Les besoins des utilisateurs

Description de ce qui sera vu par les acteurs du système. Cela correspond aux besoins attendus par chaque acteur (QUOI / QUI).

Diagramme de cas d'utilisation (CU)
Description des cas d'utilisation

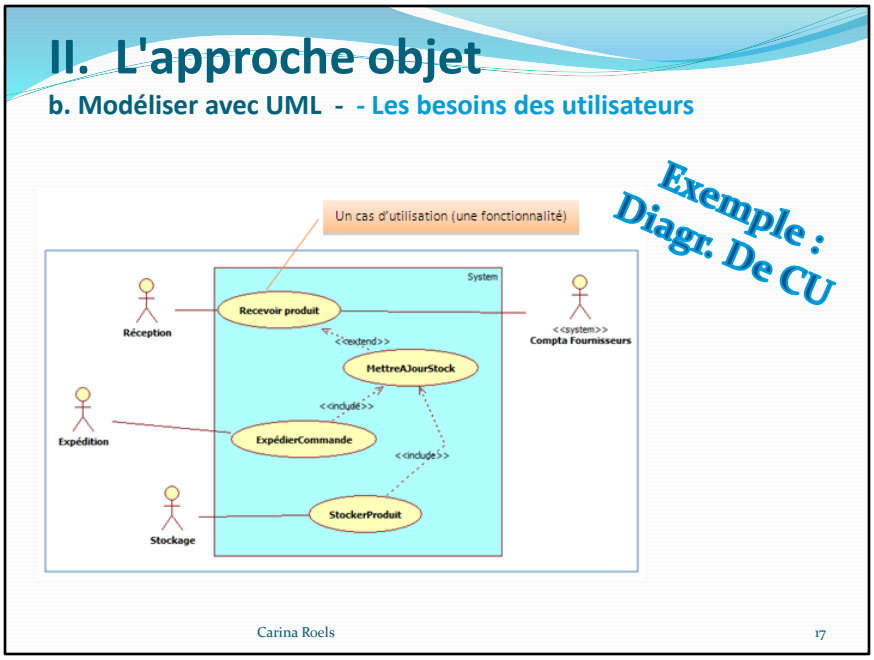


Diagramme des cas d'utilisation (Use Case Diagram) :
Identification des interactions entre le **système** et les **acteurs** (intervenants extérieurs au système) → les **fonctionnalités** que doit fournir le système.

Le diagramme de cas d'utilisation est la base pour la compréhension des besoins utilisateur.

Il servira de guide pour la réalisation et la vérification des autres vues.
Il permet également d'identifier les interfaces qui mériteraient une attention particulière.

II. L'approche objet

b. Modéliser avec UML - Les besoins des utilisateurs

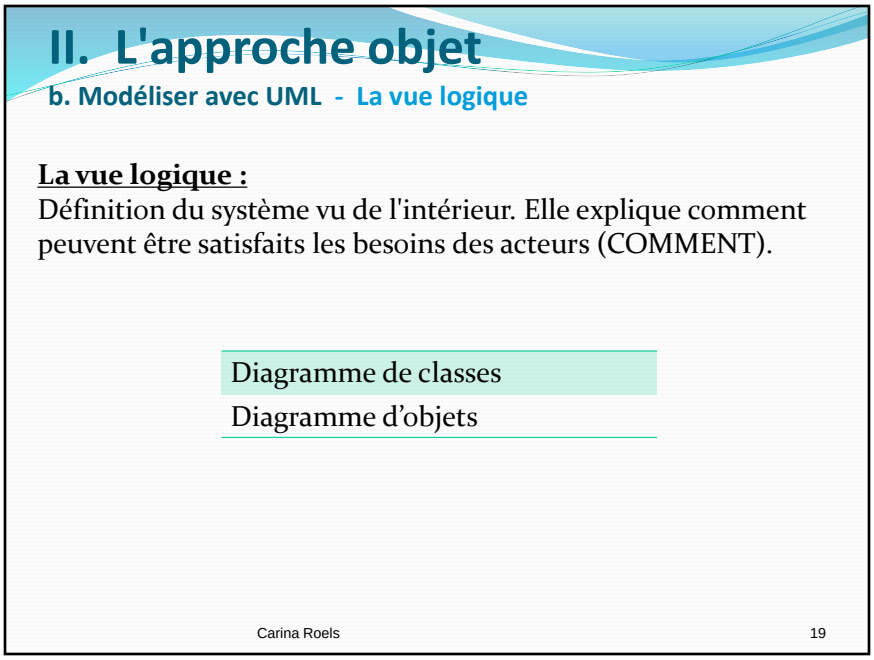
Pour chaque CU
du diagramme

Aperçu : Description de CU

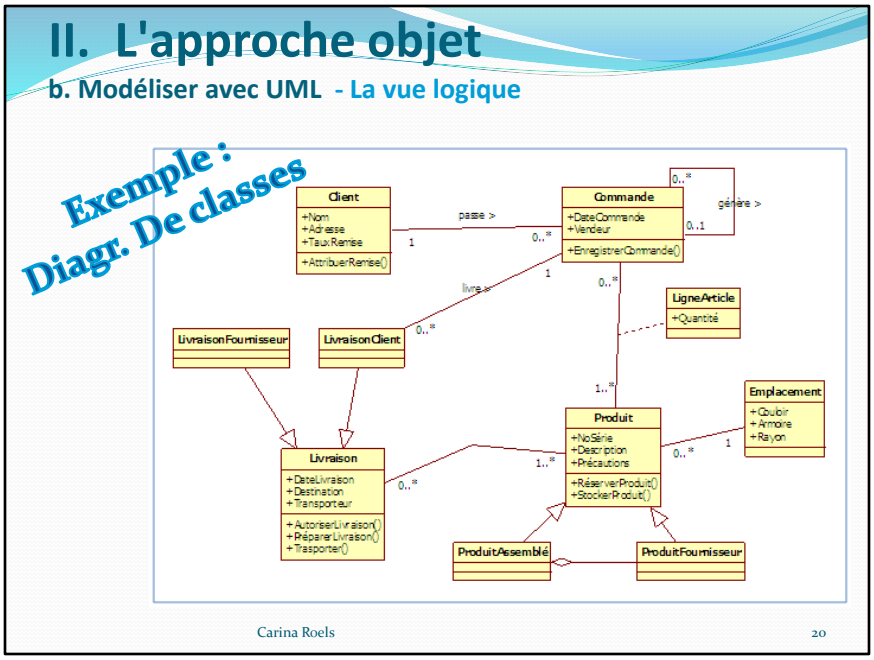
Nom _____
Présupposés _____
Pré-conditions _____

Dialogue
_____
_____
_____

Post-conditions _____
Exceptions _____
Améliorations futures
_____
Problèmes non résolus
_____

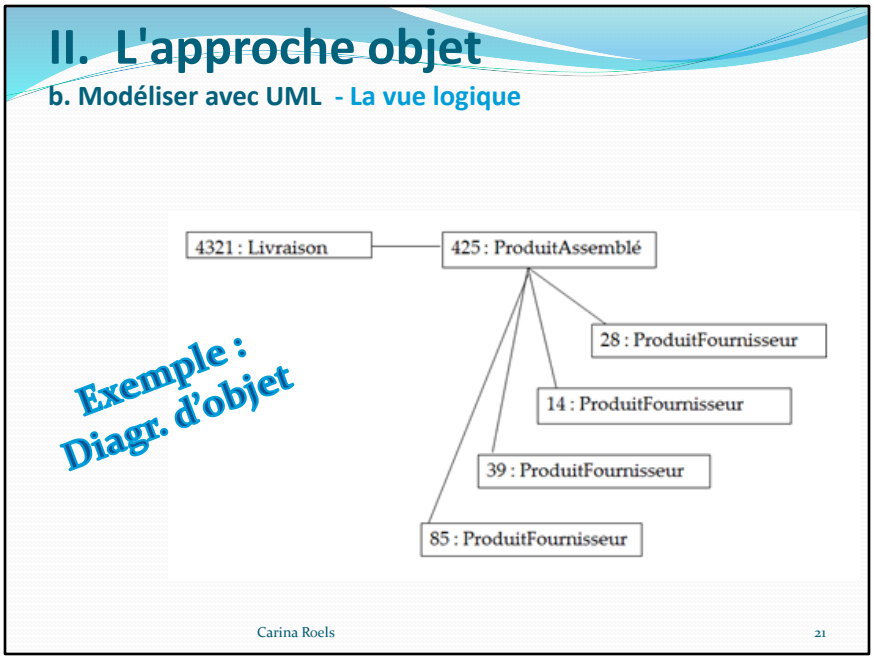


La vue logique a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments. Cette vue organise les éléments du domaine en « catégories ». Deux diagrammes peuvent être utilisés pour cette vue.



Le diagramme de classes

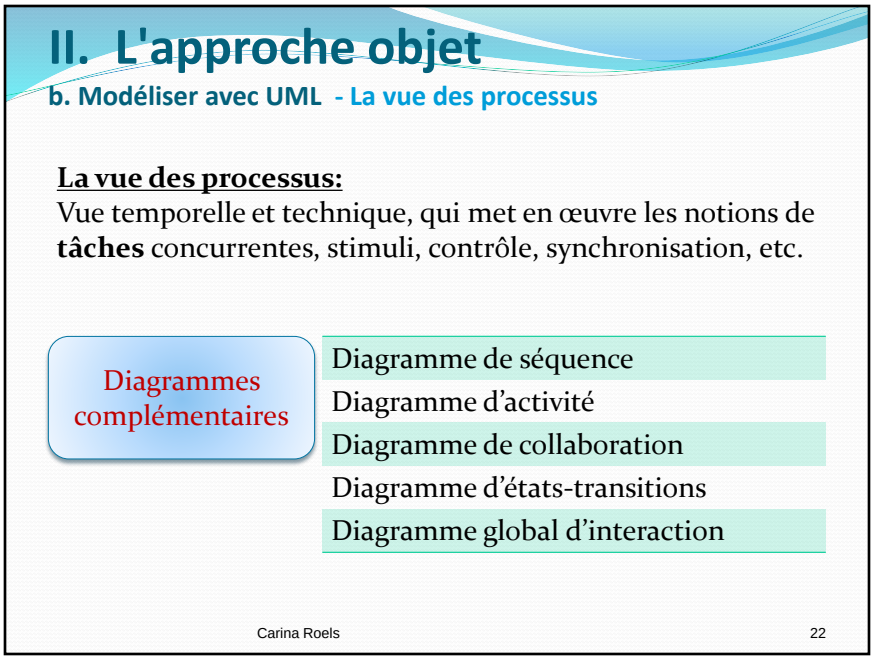
Dans la phase d'analyse, ce diagramme représente les entités (des informations) manipulées par les utilisateurs. Dans la phase de conception, il représente la structure objet d'un développement orienté objet.



Le diagramme d'objets sert à illustrer les classes complexes en utilisant des exemples d'instances.

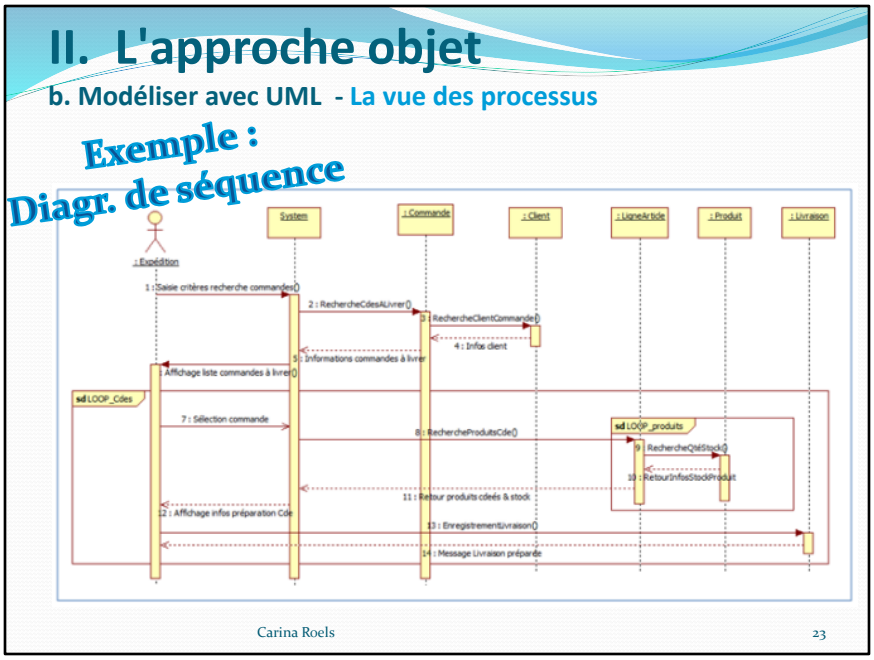
Une instance est un exemple concret de contenu d'une classe.

En illustrant une partie des classes avec des exemples (grâce à un diagramme d'objets), on arrive à voir un peu plus clairement les liens nécessaires entre les différentes classes.

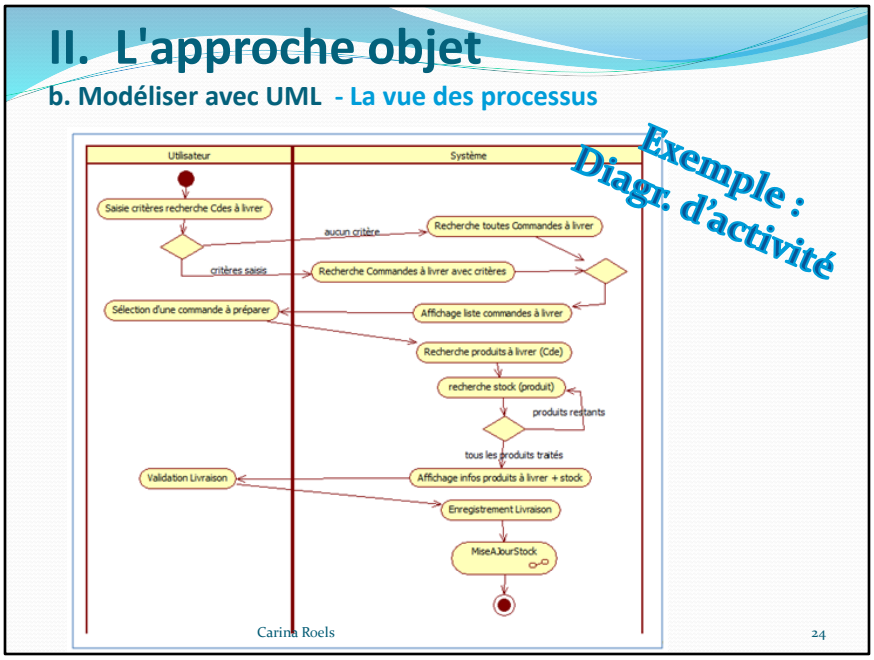


La vue des processus démontre :

- la décomposition du système en processus et actions ;
- les interactions entre les processus ;
- la synchronisation et la communication des activités parallèles.
- La vue des processus s'appuie sur plusieurs diagrammes.



Un diagramme de séquence est une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs. Il est utilisé pour compléter la description d'un cas d'utilisation.

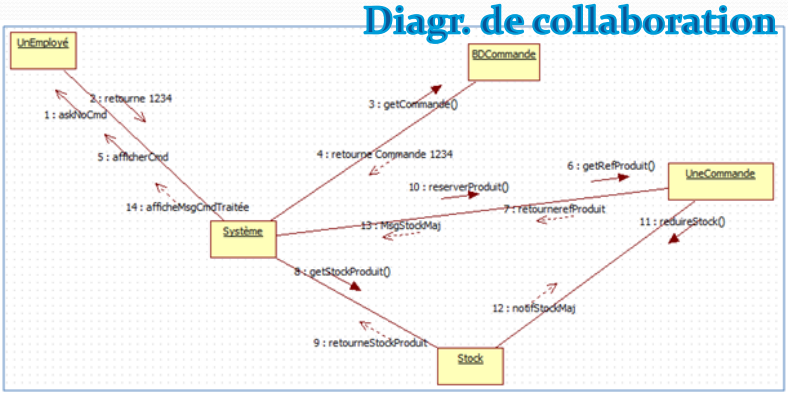


Le diagramme d'activité représente le déroulement des actions, sans utiliser les objets. En phase d'analyse, il est utilisé pour consolider les spécifications d'un cas d'utilisation.

II. L'approche objet

b. Modéliser avec UML - La vue des processus

Exemple : Diagr. de collaboration



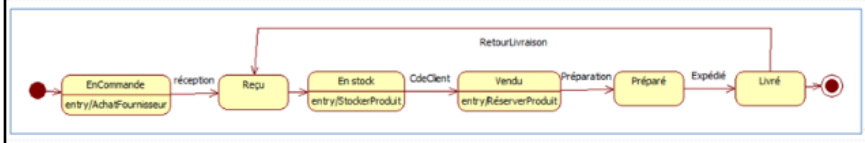
Carina Roels 25

Le diagramme de collaboration (appelé également diagramme de communication) permet de mettre en évidence les échanges de messages entre objets. Cela nous aide à voir clair dans les actions qui sont nécessaires pour produire ces échanges de messages. Et donc de compléter, si besoin, les diagrammes de séquence et de classes.

II. L'approche objet

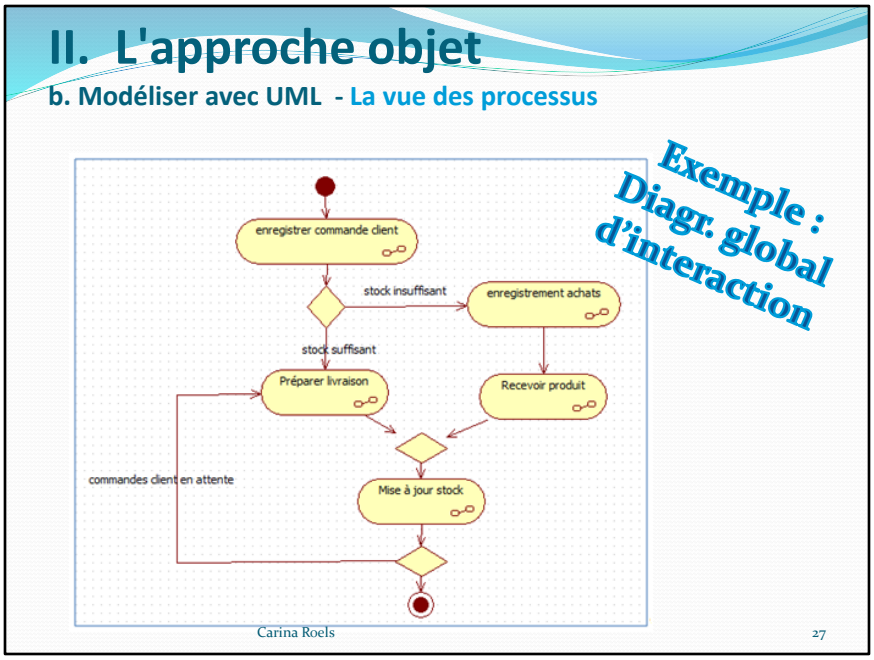
b. Modéliser avec UML - La vue des processus

Exemple :
Diagr. d'état-transition



Carina Roels 26

Le diagramme d'états-transition permet de décrire l'évolution de l'état d'un objet ou d'un composant.



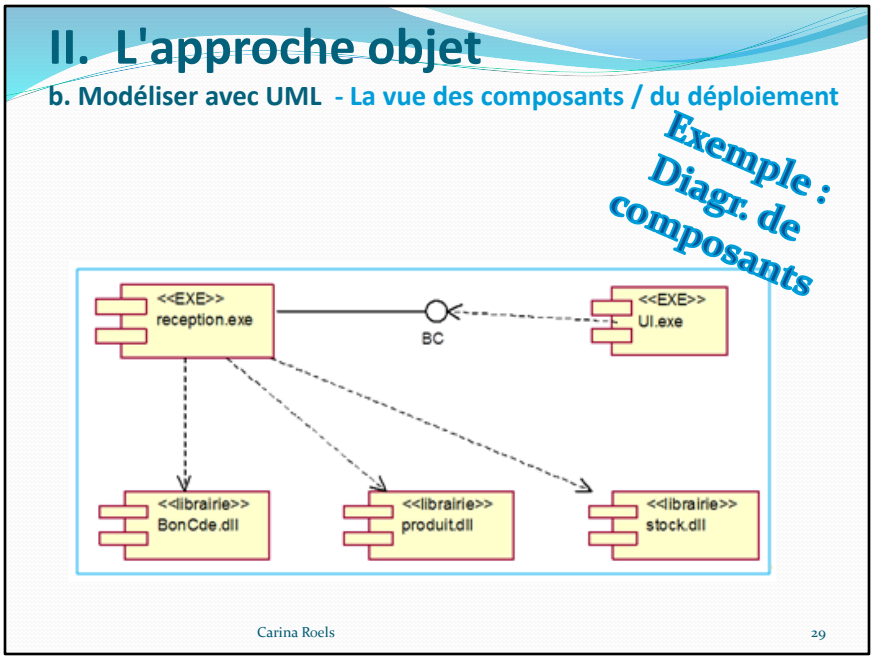
Le diagramme global d'interaction permet de donner une vue d'ensemble des interactions du système. Il est réalisé avec le même graphisme que le diagramme d'activité. Chaque élément du diagramme peut ensuite être détaillé à l'aide d'un diagramme de séquence ou d'un diagramme d'activité. Ce diagramme ne sera pas étudié dans ce cours.

II. L'approche objet

b. Modéliser avec UML - La vue des composants / du déploiement

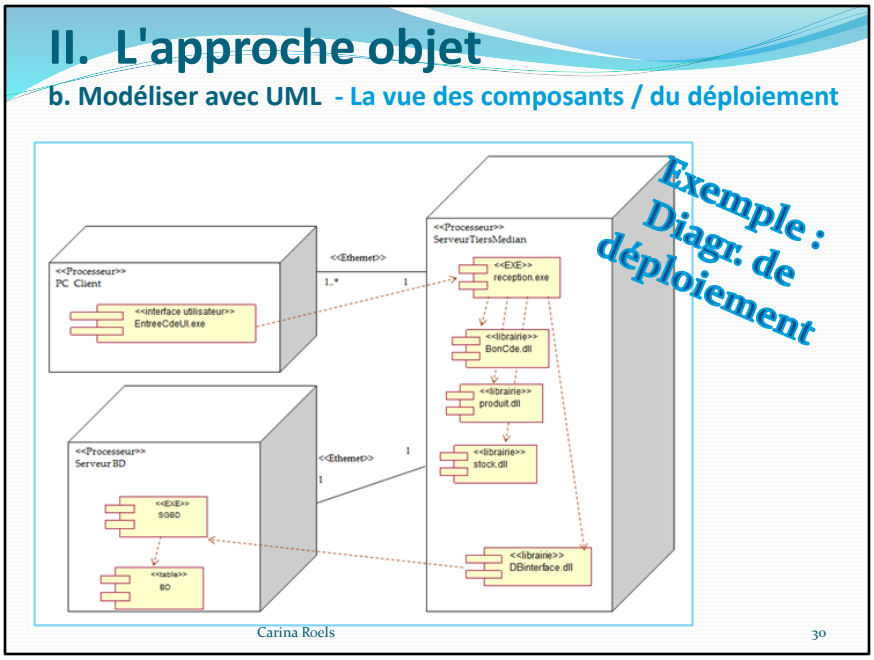
Diagrammes vus en 3ième année

- Diagramme de composants
- Diagramme de déploiement



La vue des composants (vue de réalisation) met en évidence les différentes parties qui composeront le futur système (fichiers sources, bibliothèques, bases de données, exécutables, etc.). Cette vue comprend deux diagrammes.

Le diagramme de composants décrit tous les composants utiles à l'exécution du système (applications, librairies, instances de base de données, exécutables, etc.).



La vue de déploiement décrit les ressources matérielles et la répartition des parties du logiciel sur ces éléments. Il contient un diagramme :
Le diagramme de déploiement correspond à la description de l'environnement d'exécution du système (matériel, réseau…) et de la façon dont les composants y sont installés.