COURS UML

Unified Modeling Language

Modélisation d'un Système de

GESTION DE GARAGE AUTOMOBILE

Diagrammes de Cas d'Utilisation, Séquence et Classes

Transformation vers le Modèle Entité-Relation

Auteur

Dr. BADR EL KHALYLY

Année Universitaire 2024-2025

Table des matières

Chapitre 1

Introduction à UML

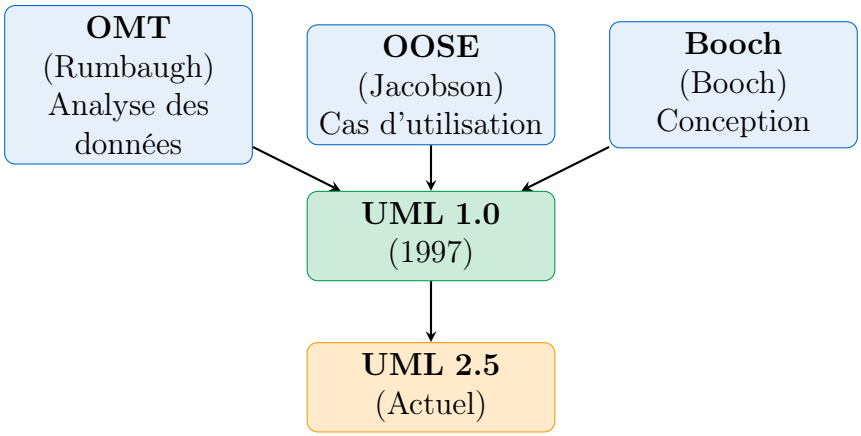
1.1 Qu'est-ce que UML ?

Définition

UML (Unified Modeling Language) est un langage de modélisation graphique standardisé, utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système logiciel.

1.1.1 Historique et Origine

UML est né de la fusion de trois méthodes orientées objet majeures dans les années 1990 :



1.1.2 Les Trois Pères Fondateurs

Auteur	Méthode	Contribution
Grady Booch	Méthode Booch	Conception et implémentation
James Rumbaugh	OMT	Analyse des données
Ivar Jacobson	OOSE	Cas d'utilisation

TABLE 1.1 – Les pères fondateurs d'UML

1.2 Les Catégories de Diagrammes UML

UML 2.5 définit **14 types de diagrammes** répartis en deux catégories :

DIAGRAMMES STRUCTURELS (7)	DIAGRAMMES COMPORTEMENTAUX (7)
• Diagramme de classes	• Diagramme de cas d'utilisation
• Diagramme d'objets	• Diagramme d'activités
• Diagramme de composants	• Diagramme d'états-transitions
• Diagramme de déploiement	• Diagramme de séquence
• Diagramme de packages	• Diagramme de communication
• Diagramme de structures composites	• Diagramme de temps
• Diagramme de profils	• Diagramme global d'interaction

1.3 Diagrammes Étudiés dans ce Cours

Dans ce cours, nous nous concentrerons sur **trois diagrammes fondamentaux** :

1. **Diagramme de Cas d'Utilisation** : répond à *QUOI ?* (Fonctionnalités)
2. **Diagramme de Séquence** : répond à *COMMENT ?* (Interactions)
3. **Diagramme de Classes** : répond à *AVEC QUOI ?* (Structure)

Chapitre 2

Présentation du Cas d'Étude

2.1 Contexte : Garage Automobile

Le garage **"AutoService Pro"** souhaite informatiser sa gestion. Voici la description du fonctionnement :

Description du Système

Le garage reçoit des **clients** qui peuvent être des particuliers ou des entreprises. Chaque client possède un ou plusieurs **véhicules**. Lorsqu'un client arrive au garage, il est accueilli par un **réceptionniste** qui enregistre sa demande d'intervention.
Le **chef d'atelier** analyse la demande et affecte un ou plusieurs **mécaniciens** pour effectuer la **réparation**. Pendant l'intervention, les mécaniciens peuvent utiliser des **pièces de rechange** du stock.
Une fois la réparation terminée, une **facture** est générée. Le client peut payer en espèces, par carte ou par chèque. Le système doit également permettre de gérer le **stock de pièces** et de consulter l'**historique des interventions**.

2.2 Analyse du Texte

2.2.1 Identification des Éléments

Catégorie	Éléments identifiés	Source dans le texte
Acteurs	Client	"Le garage reçoit des clients"
	Réceptionniste	"accueilli par un réceptionniste"
	Chef d'atelier	"Le chef d'atelier analyse"
	Mécanicien	"affecte des mécaniciens"
Entités	Véhicule	"possède des véhicules"
	Intervention	"demande d'intervention"
	Pièce	"pièces de rechange"
	Facture	"une facture est générée"
	Stock	"stock de pièces"
Actions	Enregistrer intervention	"enregistre sa demande"
	Affecter mécanicien	"affecte des mécaniciens"
	Effectuer réparation	"effectuer la réparation"
	Générer facture	"facture est générée"
	Payer	"Le client peut payer"

TABLE 2.1 – Extraction des éléments du cahier des charges

Chapitre 3

Diagramme des Cas d'Utilisation

3.1 Définition et Objectifs

Définition

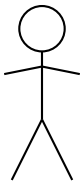
Le **diagramme de cas d'utilisation** (Use Case Diagram) représente les fonctionnalités d'un système du point de vue de l'utilisateur. Il répond à la question : **"QUE fait le système ?"**

3.2 Éléments Constitutifs

3.2.1 Les Acteurs

Acteur

Un **acteur** représente un rôle joué par une personne, un système externe ou un dispositif qui interagit avec le système étudié.

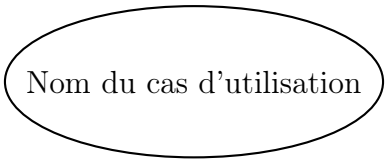


Nom de l'acteur

3.2.2 Les Cas d'Utilisation

Cas d'Utilisation

Un **cas d'utilisation** représente une fonctionnalité ou un service que le système offre à ses utilisateurs pour atteindre un objectif.



3.3 Les Relations entre Cas d'Utilisation

Relations Importantes

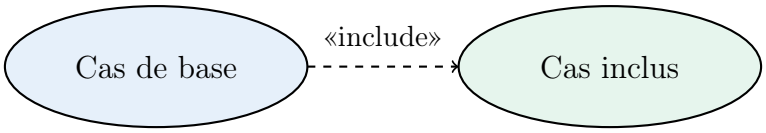
Il existe trois types de relations entre cas d'utilisation :

1. **Association** : lien simple entre acteur et cas d'utilisation
2. **Include** : inclusion obligatoire
3. **Extend** : extension optionnelle

3.3.1 La Relation «include»

Définition de «include»

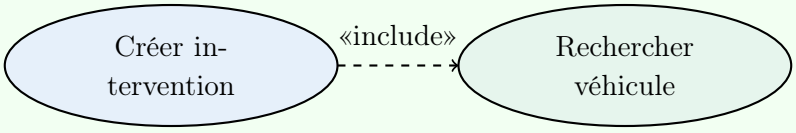
La relation **«include»** indique qu'un cas d'utilisation (cas de base) **inclut obligatoirement** le comportement d'un autre cas d'utilisation (cas inclus).



Exemple Concret - Garage

Situation : Pour créer une intervention, on DOIT obligatoirement rechercher le véhicule concerné.

Modélisation :



Justification :

- Le texte dit "sa demande d'intervention" ⇒ on doit identifier LE véhicule
- Sans véhicule identifié, impossible de créer l'intervention
- La recherche est donc **obligatoire** = «include»

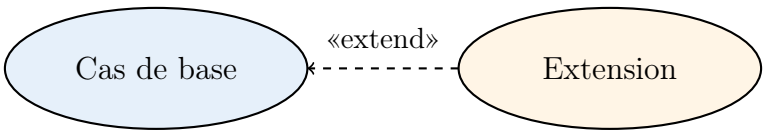
Quand utiliser «include» ?

- Le comportement inclus est **obligatoire**
- Le comportement inclus est **réutilisé** par plusieurs cas
- On veut **factoriser** un comportement commun

3.3.2 La Relation «extend»

Définition de «extend»

La relation **«extend»** indique qu'un cas d'utilisation (cas d'extension) **peut optionnellement** étendre le comportement d'un autre cas d'utilisation (cas de base), sous certaines conditions.



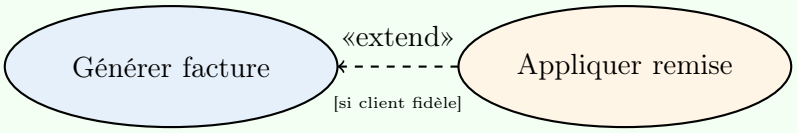
Direction de la Flèche

Attention ! La flèche «extend» va de l'extension vers le cas de base (sens inverse de «include»).

Exemple Concret - Garage

Situation : Lors de la génération d'une facture, on PEUT optionnellement appliquer une remise.

Modélisation :



Justification :

- La remise n'est PAS toujours appliquée
- C'est une option, pas une obligation
- Condition : "si client fidèle" ou "si montant $>$ 500€"
- Le cas de base (Générer facture) fonctionne SANS l'extension

Quand utiliser «extend» ?

- Le comportement est **optionnel**
- Il y a une **condition** pour déclencher l'extension
- Le cas de base est **complet** sans l'extension

3.3.3 Comparaison «include» vs «extend»

Critère	**«include»**	**«extend»**
Obligation	Obligatoire (toujours exécuté)	Optionnel (sous condition)
Direction flèche	Base → Inclus	Extension → Base
Dépendance	Base dépend de l'inclus	Extension dépend de la base
Complétude	Base incomplet sans l'inclus	Base complet sans l'extension
Exemple garage	Créer intervention → Rechercher véhicule	Appliquer remise → Générer facture

TABLE 3.1 – Comparaison entre «include» et «extend»

3.4 Diagramme de Cas d'Utilisation Complet

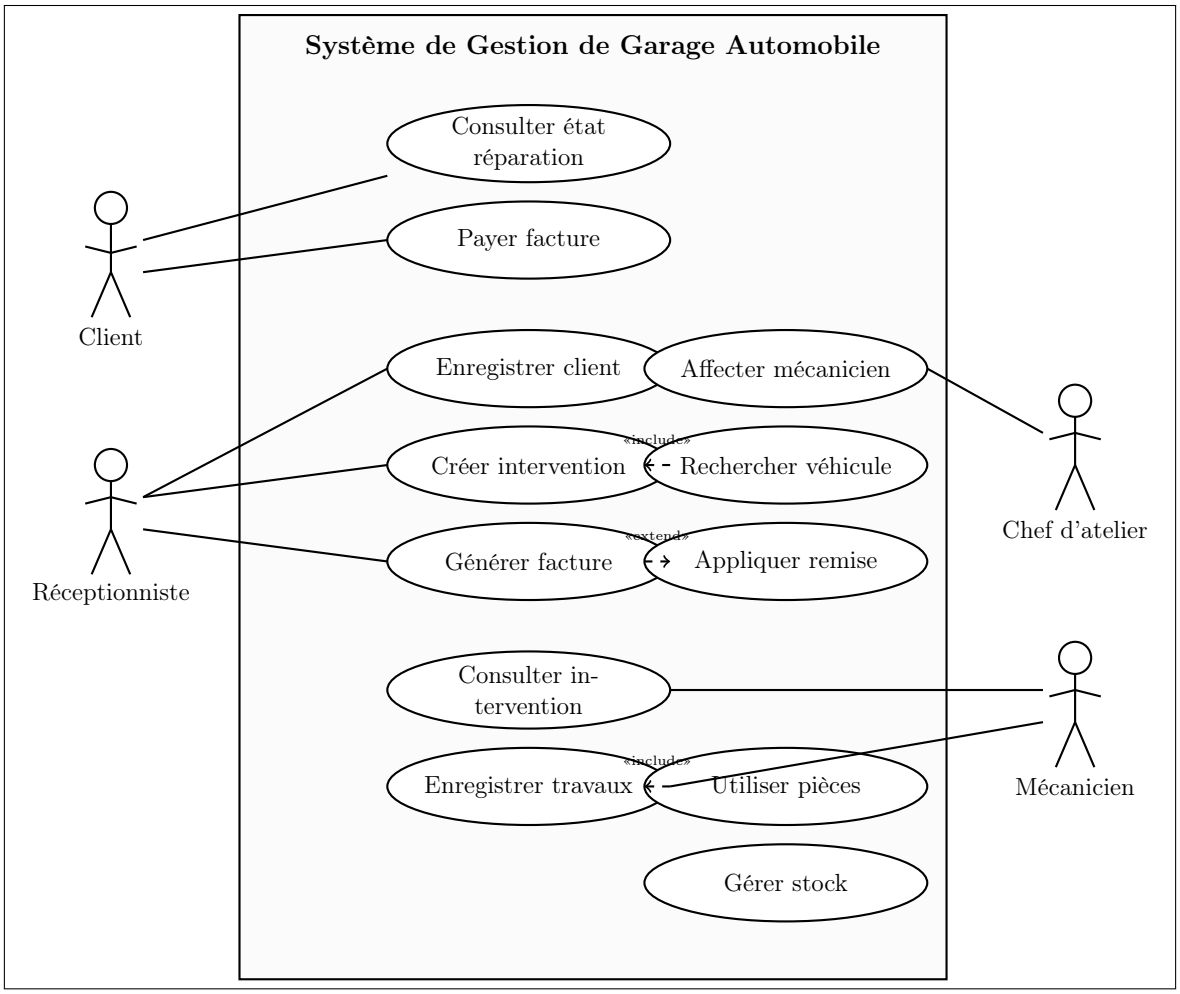


FIGURE 3.1 – Diagramme de Cas d'Utilisation - Garage Automobile

Chapitre 4

Diagrammes de Séquence

4.1 Définition et Objectifs

Définition

Le **diagramme de séquence** représente les interactions entre objets en mettant l'accent sur l'ordre chronologique des messages échangés. Il répond à la question : **"COMMENT se déroulent les interactions dans le temps ?"**

4.2 Éléments Constitutifs

Élément	**Description**	**Notation**
Participant	Objet qui participe à l'interaction	Rectangle en haut
Ligne de vie	Durée de vie de l'objet	Ligne verticale pointillée
Message synchrone	Appel avec attente de réponse	Flèche pleine →
Message retour	Réponse à un message	Flèche pointillée --→
Barre d'activation	Période d'activité	Rectangle sur ligne de vie
Fragment alt	Alternative (if-else)	Cadre avec "alt"
Fragment loop	Boucle	Cadre avec "loop"
Fragment ref	Référence (include)	Cadre avec "ref"

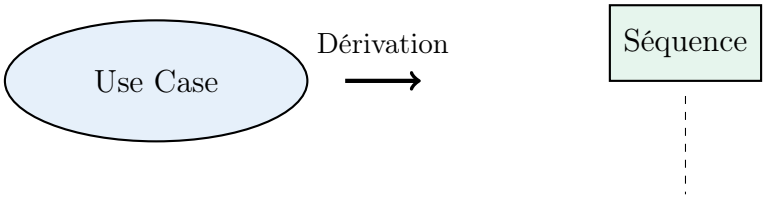
TABLE 4.1 – Éléments d'un diagramme de séquence

4.3 Dérivation Use Case vers Séquence

Principe de Dérivation

Chaque cas d'utilisation peut être détaillé par un ou plusieurs diagrammes de séquence :

- Un diagramme pour le **scénario nominal** (cas normal)
- Un diagramme pour chaque **scénario alternatif** (exceptions)



4.4 Diagramme de Séquence : Enregistrer Client (UC-01)

4.4.1 Provenance et Justification

Use Case	UC-01 : Enregistrer nouveau client
Acteur	Réceptionniste
Précondition	Réceptionniste authentifié
Postcondition	Client enregistré dans le système

4.4.2 Identification des Participants

- **:Réceptionniste** ← Acteur du Use Case
- **:IHM** ← Interface utilisateur (couche présentation)
- **:CtrlClient** ← Contrôleur métier (logique)
- **:BD** ← Base de données (persistance)

4.4.3 Diagramme - Scénario Nominal

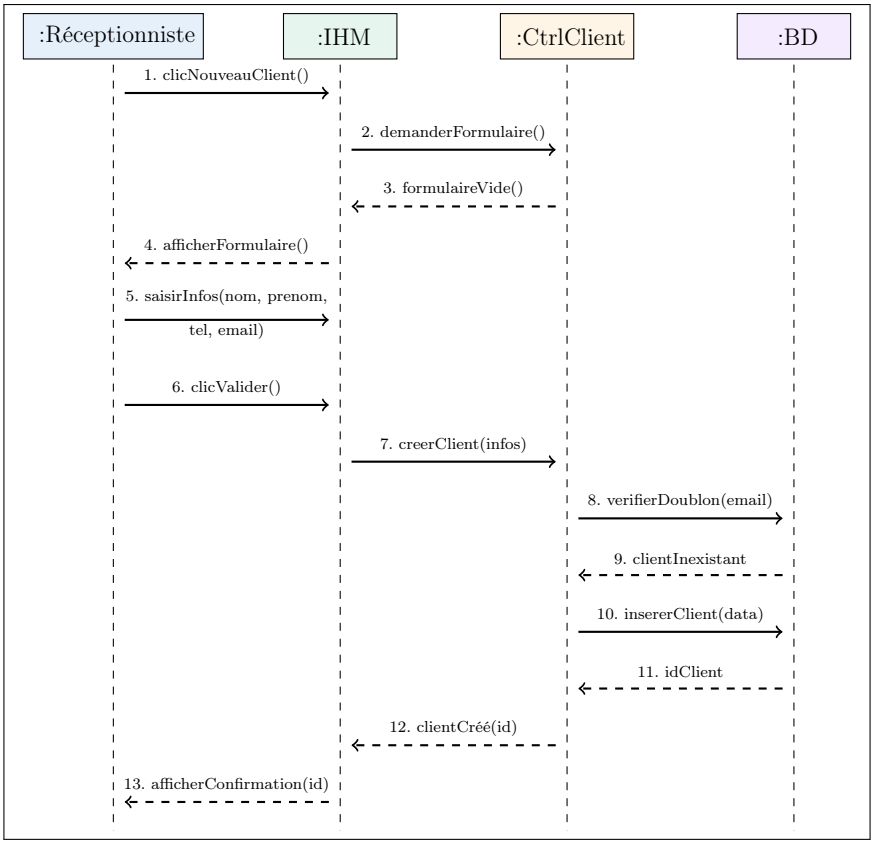


FIGURE 4.1 – Diagramme de Séquence - Enregistrer Client (Scénario Nominal)

4.5 Diagramme de Séquence : Créer Intervention (UC-02)

4.5.1 Traitement du «include»

Représentation du «include» en Séquence

Lorsqu'un Use Case inclut un autre Use Case via «include», le diagramme de séquence représente cela par un **fragment "ref"** (référence) qui encadre les messages correspondants.

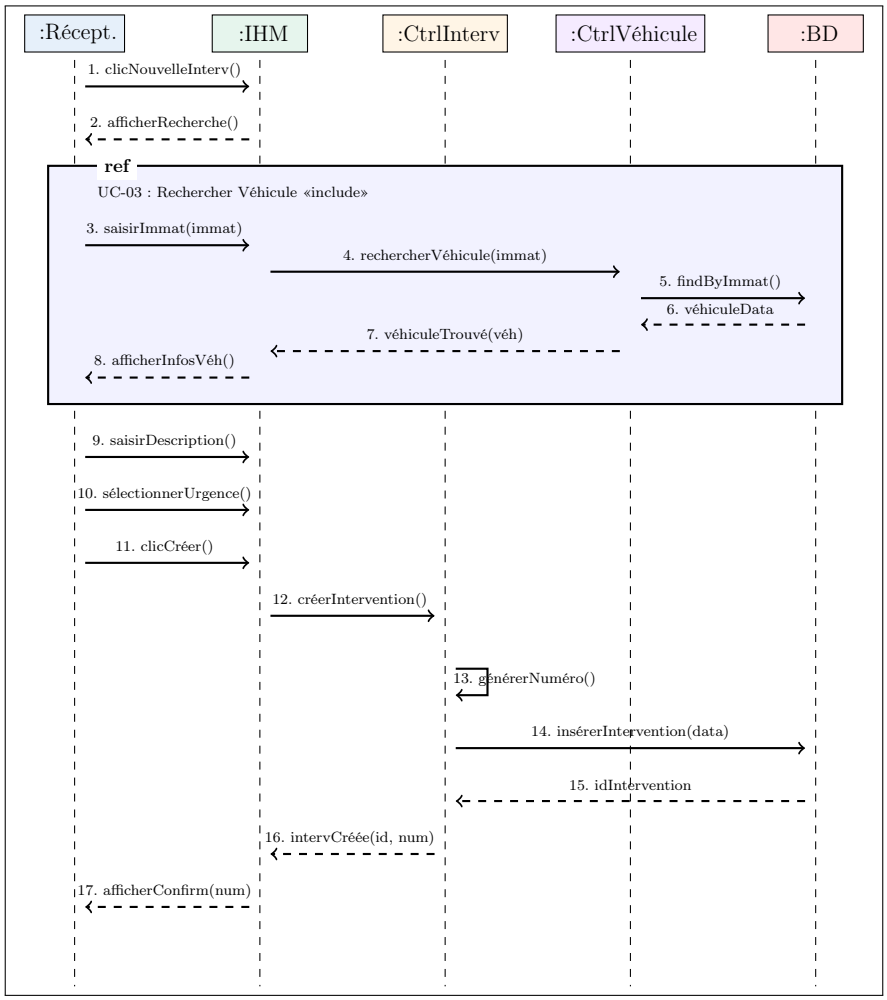


FIGURE 4.2 – Diagramme de Séquence - Créer Intervention avec «include»

4.6 Diagramme de Séquence : Effectuer Réparation (UC-11)

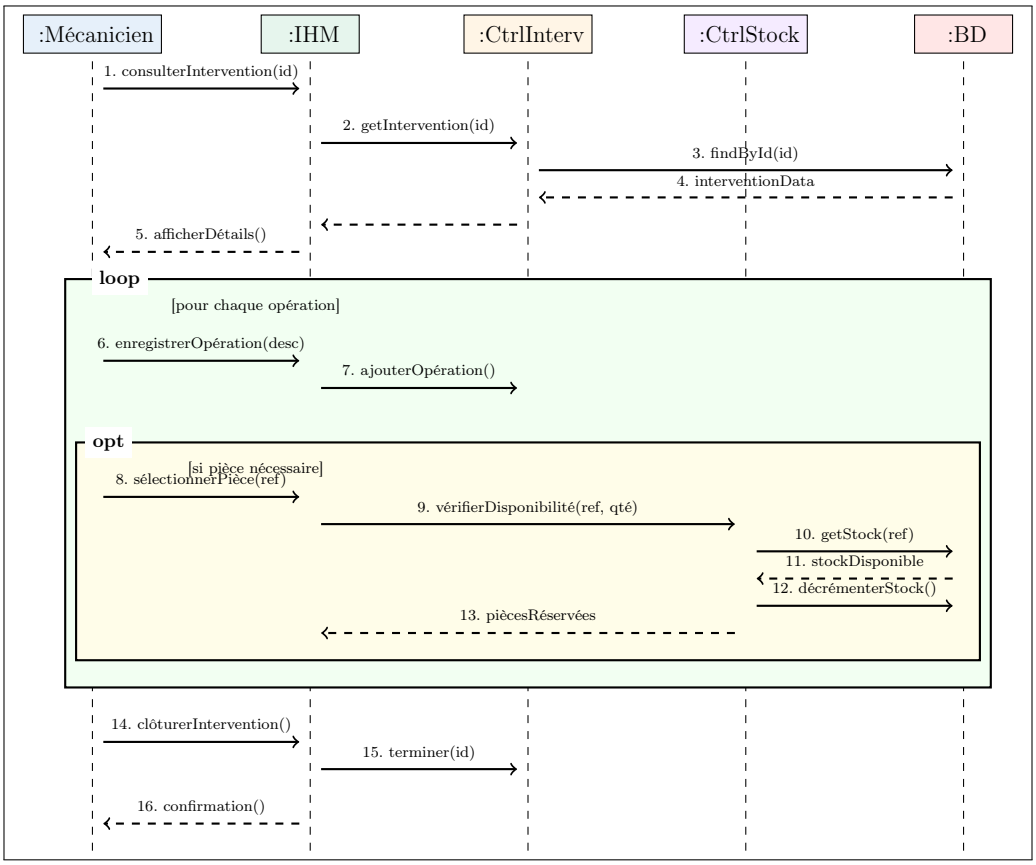


FIGURE 4.3 – Diagramme de Séquence - Effectuer Réparation

4.7 Diagramme de Séquence : Générer Facture (UC-04)

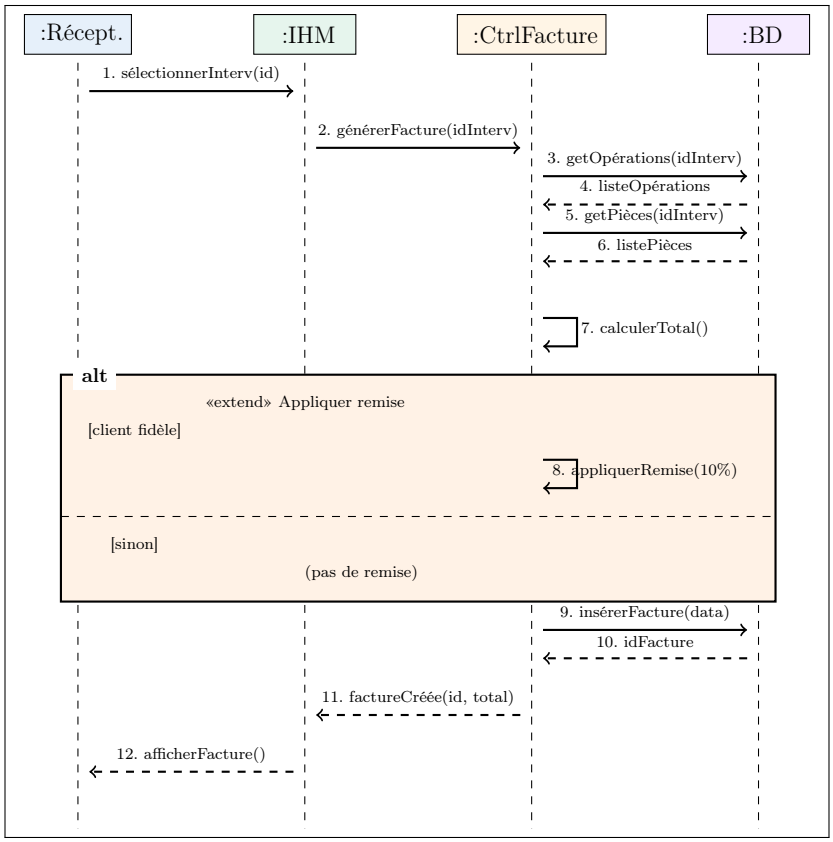


FIGURE 4.4 – Diagramme de Séquence - Générer Facture avec «extend»

Représentation du «extend» en Séquence

La relation «extend» se traduit par un **fragment "alt"** (alternative) dans le diagramme de séquence, car l'extension est **conditionnelle**.

Chapitre 5

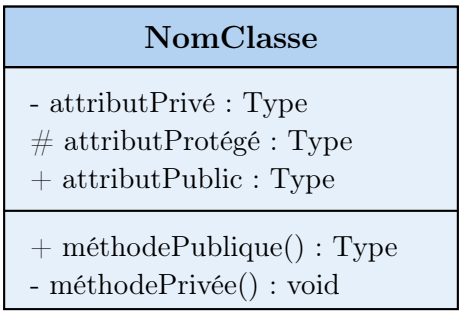
Diagramme de Classes

5.1 Définition et Objectifs

Définition

Le **diagramme de classes** représente la structure statique du système en montrant les classes, leurs attributs, leurs méthodes et les relations entre elles. Il répond à la question : **"AVEC QUOI le système est-il construit ?"**

5.2 Structure d'une Classe



5.2.1 Visibilité des Membres

Symbole	Visibilité	Accès
+	public	Accessible par tous
-	private	Accessible uniquement dans la classe
#	protected	Accessible dans la classe et ses sous-classes
~	package	Accessible dans le même package

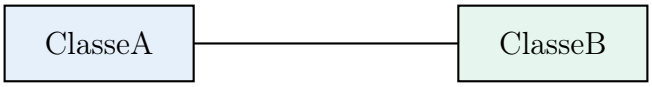
TABLE 5.1 – Symboles de visibilité UML

5.3 Les Types d'Associations

5.3.1 Association Simple

Association Simple

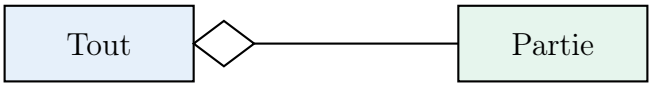
Une **association** représente un lien structurel entre deux classes. Elle indique que les instances d'une classe sont connectées aux instances d'une autre classe.



5.3.2 Agrégation

Agrégation (losange vide)

L'**agrégation** est une association "partie-tout" **faible**. La partie peut exister indépendamment du tout.



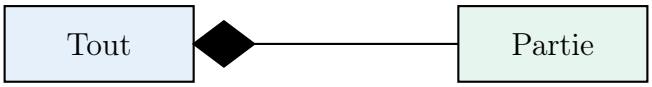
Exemple - Garage

Un **Garage** agrège des **Mécaniciens**. Si le garage ferme, les mécaniciens existent toujours (ils peuvent travailler ailleurs).

5.3.3 Composition

Composition (losange plein)

La **composition** est une agrégation **forte**. La partie ne peut PAS exister sans le tout. Si le tout est détruit, la partie l'est aussi.



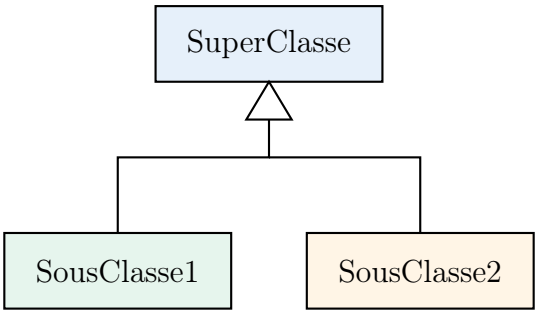
Exemple - Garage

Une **Facture** est composée de **LignesFacture**. Si la facture est supprimée, ses lignes n'ont plus de sens et sont supprimées aussi.

5.3.4 Héritage / Généralisation

Héritage

L'**héritage** (ou généralisation) permet à une classe (sous-classe) d'hériter des attributs et méthodes d'une autre classe (super-classe).



5.4 Les Multiplicités (Cardinalités)

Multiplicités - Définitions

Les multiplicités indiquent combien d'instances d'une classe peuvent être associées à une instance d'une autre classe.

Notation	Signification	Exemple
1	Exactement un	Un véhicule a 1 propriétaire
0..1	Zéro ou un	Un employé peut avoir 0 ou 1 bureau
* ou 0..*	Zéro ou plusieurs	Un client peut avoir 0 à n véhicules
1..*	Un ou plusieurs	Une facture a au moins 1 ligne
n..m	Entre n et m	Une équipe a entre 2 et 5 membres

TABLE 5.2 – Notation des multiplicités

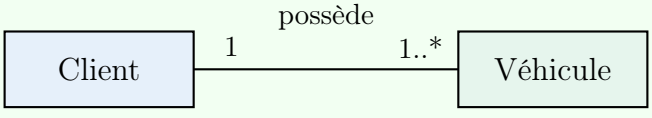
5.4.1 One-to-Many (1..*) vs Many-to-One (*..1)

Comprendre les Cardinalités

La notation se lit **du côté opposé**. Pour une association A — B :

- La multiplicité près de B indique : "Pour chaque A, combien de B ?"
- La multiplicité près de A indique : "Pour chaque B, combien de A ?"

Exemple 1 : Client - Véhicule (One-to-Many)

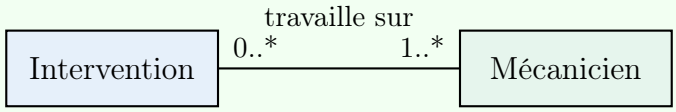


Lecture :

- Un Client possède **1 ou plusieurs** Véhicules (1..* côté Véhicule)
- Un Véhicule appartient à **exactement 1** Client (1 côté Client)

C'est One-to-Many car : UN client → PLUSIEURS véhicules

Exemple 2 : Intervention - Mécanicien (Many-to-Many)



Lecture :

- Une Intervention nécessite **1 ou plusieurs** Mécaniciens
- Un Mécanicien peut travailler sur **0 ou plusieurs** Interventions

C'est Many-to-Many car : PLUSIEURS interventions ↔ PLUSIEURS mécaniciens

5.5 Diagramme de Classes Complet - Garage

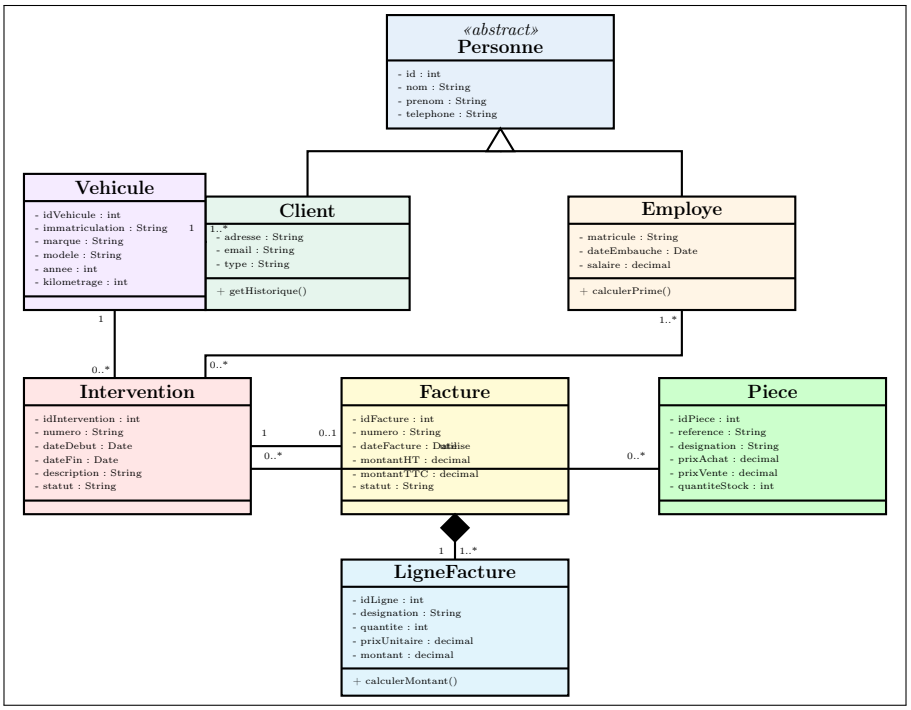


FIGURE 5.1 – Diagramme de Classes - Système de Gestion de Garage

Chapitre 6

Transformation vers le Modèle Entité-Relation

6.1 Correspondance UML - Entité-Relation

UML	E/R	SQL
Classe	Entité	Table
Attribut	Attribut	Colonne
Identifiant	Identifiant	Clé primaire (PK)
Association	Relation	Clé étrangère (FK)
Héritage	Spécialisation	3 stratégies possibles

TABLE 6.1 – Correspondance entre UML, E/R et SQL

6.2 Règles de Transformation des Associations

6.2.1 Règle 1 : Association One-to-One (1..1)

Transformation 1 :1

La clé étrangère peut être placée dans **l'une ou l'autre** des tables. On choisit généralement le côté le plus "dépendant".

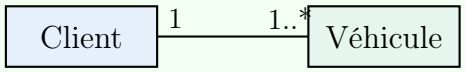
6.2.2 Règle 2 : Association One-to-Many (1..*)

Transformation 1 :N

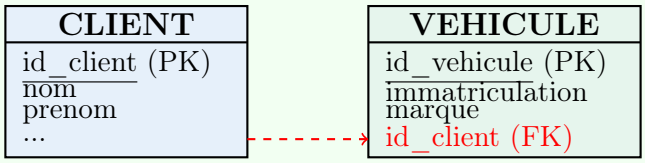
La clé étrangère est placée **du côté "Many"** (N).

Exemple : Client - Véhicule

UML :



SQL :



Justification : La FK est dans VEHICULE car un véhicule appartient à UN client (côté Many).

6.2.3 Règle 3 : Association Many-to-Many (*..*)

Transformation M :N

Création d'une **table d'association** (table de jonction) contenant les clés étrangères des deux tables.

Exemple : Intervention - Mécanicien

La table AFFECTATION :

- Contient les FK des deux tables liées
- Sa PK est composée des deux FK
- Peut contenir des attributs propres (date_affectation)

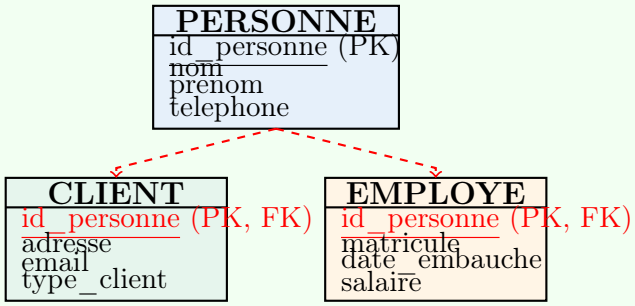
6.3 Transformation de l'Héritage

Trois Stratégies Possibles

1. **Single Table** : Une seule table pour toute la hiérarchie
2. **Table per Class** : Une table par classe concrète
3. **Joined** : Une table par classe (recommandé)

6.3.1 Stratégie Joined (Recommandée)

Transformation de Personne/Client/Employe



Caractéristiques :

- La PK des sous-tables est aussi FK vers la table parente
- Pas de redondance des attributs communs
- Nécessite une jointure pour reconstituer l'objet complet

6.4 Schéma Entité-Relation Complet

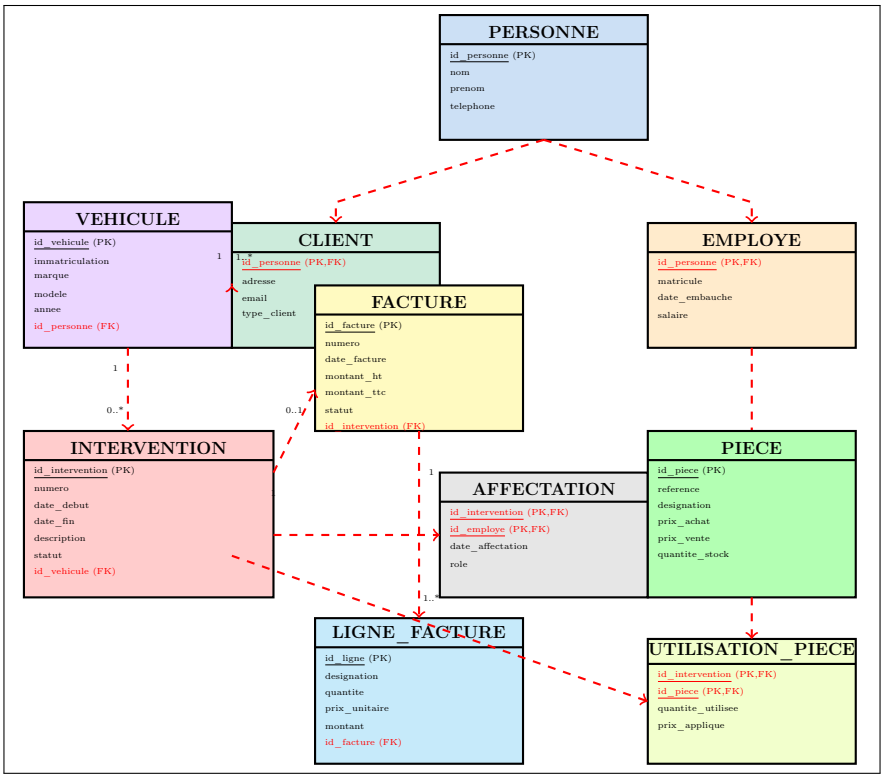
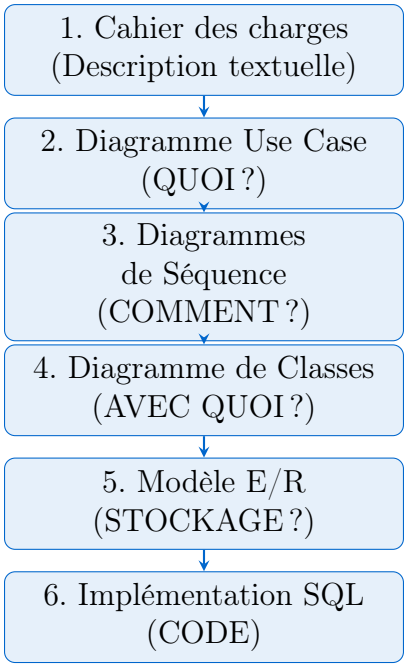


FIGURE 6.1 – Modèle Entité-Relation Complet - Garage Automobile

Chapitre 7

Conclusion

7.1 Récapitulatif de la Démarche



7.2 Points Clés à Retenir

Diagramme	**Question**	**Éléments clés**
Use Case	QUOI ?	Acteurs, Cas, Include, Extend
Séquence	COMMENT ?	Objets, Messages, Fragments
Classes	AVEC QUOI ?	Classes, Attributs, Associations
E/R	STOCKAGE ?	Entités, Clés, Relations

TABLE 7.1 – Résumé des diagrammes étudiés

7.3 Bibliographie

— **Rumbaugh, J., Jacobson, I., Booch, G.** (2004). *The Unified Modeling Language Reference Manual*, 2nd Edition. Addison-Wesley.

— **Fowler, M.** (2003). *UML Distilled : A Brief Guide to the Standard Object Modeling Language*, 3rd Edition. Addison-Wesley.

— **Larman, C.** (2004). *Applying UML and Patterns*, 3rd Edition. Prentice Hall.

— **OMG** (2017). *OMG Unified Modeling Language Specification*, Version 2.5.1.

Fin du Cours

Dr. BADR EL KHALYLY

Année Universitaire 2024-2025