Université Abdelhamid Mehri – Constantine 2

2020/2021. Semestre 5

Génie Logiciel 2

– Cours –

Chapitre 3 : Analyse

Staff pédagogique			
Nom	**Grade**	**Faculté/Institut**	**Adresse e-mail**
Dr. Sahar SMAALI	MCB	Nouvelles technologies	sahar.smaali@univ-constantine2.dz

Étudiants concernés			
Faculté/Institut	**Département**	**Niveau**	**spécialité**
Nouvelles technologies	TLSI	Licence 3	GL

Objectifs du cours

- Apprendre à mener une analyse orientée objet par élaboration des diagrammes de classes de domaine et participantes
- Apprendre à décrire un système selon le modèle MVC
- Appendre à décrire le comportement d'un système par des diagramme de séquence, activités et états/transitions

Sommaire

1 Introduction **2**

2 Modèle d'analyse **2**

3 Modele d'analyse Objet **3**
3.1 Diagramme de classes . . . 3
3.1.1 Éléments de base . . . 3
3.1.2 Relations entre classes . . . 4
3.2 Classes de domaine . . . 6
3.3 Classes d'analyse . . . 8
3.3.1 Modèle MVC . . . 8
3.3.2 Diagramme de classes participantes . . . 9

4 Modèle d'analyse dynamique **10**
4.1 Diagramme de Séquence détaillé . . . 10
4.2 Diagramme d'états/transitions . . . 12

1 Introduction

La spécification des besoins résulte en un modèle de cas d'utilisation et une maquette d'IHM permettant d'avoir une vision fonctionnelle du système. Ce modèle représente aussi un point d'entrée pour l'analyse des scénarios d'exécution du futur système et leurs réalisations. L'activité de l'analyse se fait durant la phase d'élaboration dont le but de Réaliser des spécifications des cas d'utilisation pour faciliter et orienter la conception de la solution. Elle consiste essentiellement à rechercher les objets du domaine, leurs propriétés, leurs relations ainsi que leurs interactions. Ce chapitre donne un aperçu sur l'activité d'analyse tout en adoptant le modèle MVC pour la spécification objet des cas d'utilisation.

2 Modèle d'analyse

En fait, le modèle d'analyse se compose de trois modèles individuels (voir la figure 1):

- Modèle fonctionnel (représenté par les cas d'utilisation et les scénarios) i.e. modèle des cas d'utilisation. Ce modèle peut éventuellement contenir une maquette d'IHM (Interface Homme-Machine) qui est en fait un produit jetable permettant aux utilisateurs d'avoir une vue concrète mais non définitive de la future interface de l'application.
- Modèle d'analyse objet (exprimé généralement par des diagrammes de classes et d'objets)
- Modèle dynamique (décrit souvent par des diagrammes d'interaction, d'état/transition ou encore d'activité).

Nous nous intéressons dans ce chapitre donc au raffinement du modèle des cas d'utilisation pour dériver les deux derniers modèles en suivant les sous-activités définies par le diagramme de la figure 2.

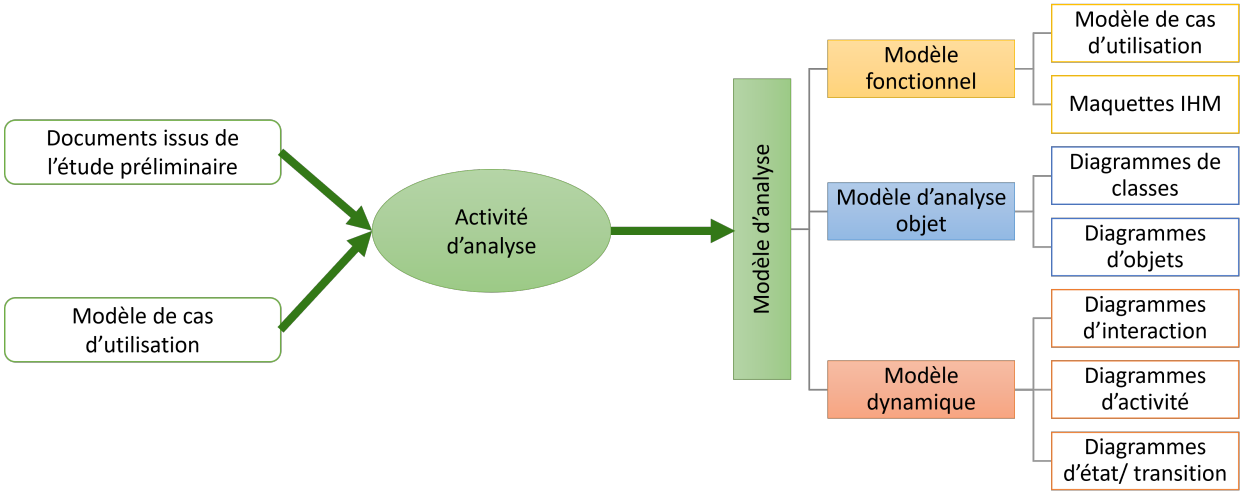

Figure 1: Modèle d'analyse.

Remarque

Il faut bien distinguer entre analyse objet et analyse des besoins dans laquelle les développeurs expriment et restructurent les besoins du point de vue des utilisateurs et du client en établissant un modèle de cas d'utilisation.
Il y a une ambiguïté entre l'analyse et la conception. Les deux répondent à la question «comment »: L'analyse se focalise sur l'aspect métier des fonctionnalités. La conception se focalise sur l'aspect technique.

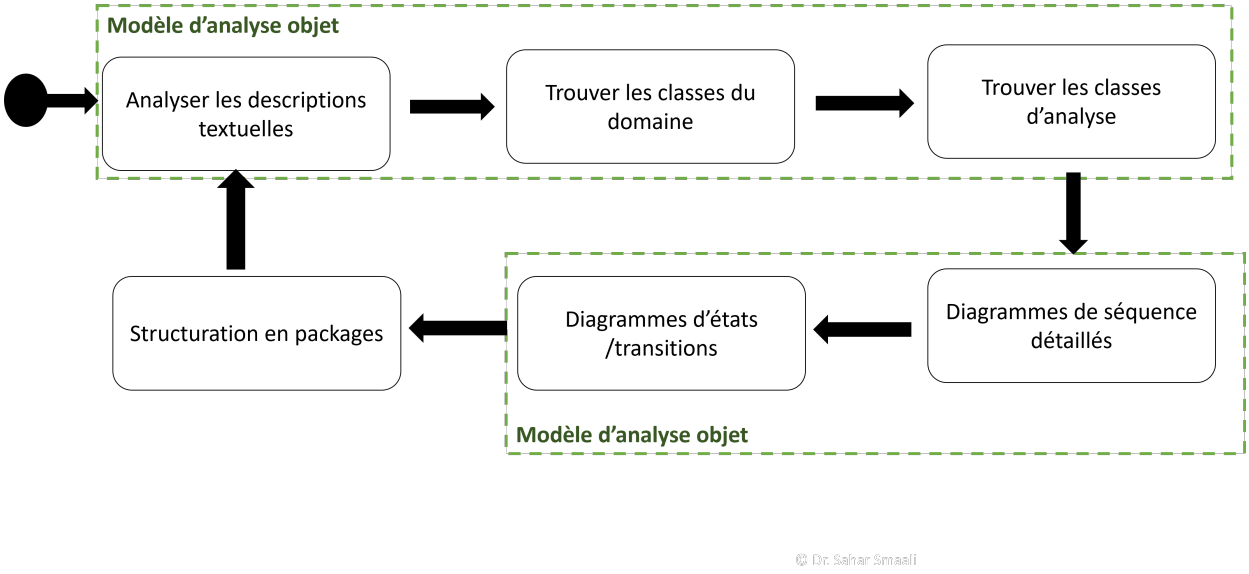

Figure 2: Démarche d'élaboration du modèle d'analyse.

3 Modele d'analyse Objet

3.1 Diagramme de classes

3.1.1 Éléments de base

Objet Un objet est une entité du monde réel possédant une identité et encapsulant un état et un comportement. Il possède un identifiant unique qui permet de le distinguer des autres objets.

Classe Une classe est un descripteur d'un ensemble d'objets qui partagent les mêmes les mêmes propriétés (attributs), un même comportement (opérations), et une sémantique commune (domaine de définition). Un objet est une instance d'une classe c.a.d. La classe est le modèle, l'objet est sa réalisation. Elle est définie par son nom, ses attributs et ses opérations comme suit :

- **Attribut** Un attribut est une propriété élémentaire d'une classe. Pour chaque objet d'une classe, l'attribut prend une valeur (sauf cas d'attributs multivalués). Un attribut est définit par le formalisme suivant:
  Visibilité nomAttribut : Type [= ValeurParDefaut]
- **Operation** Une opération est une fonction applicable aux objets d'une classe. Une opération permet de décrire le comportement d'un objet. Une méthode est l'implémentation d'une opération. Elle est définit selon le formalisme suivant:
  Visibilité nomOperation ([param1, ... , paramN]) : [typeRetour] [propriétés]

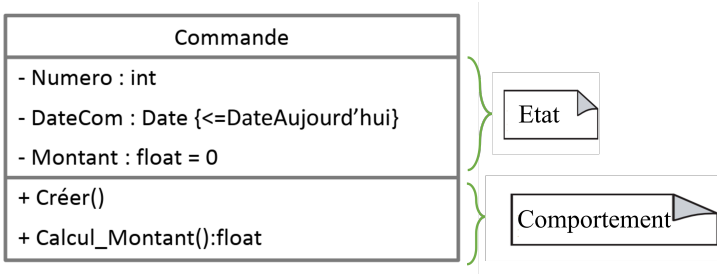

Figure 3: Représentation d'une classe

- **Visibilité** spécifie le type de visibilité autorisé pour les autres classes:
  - Public (+) : Attribut ou opération visible par tous.
  - Protégé (#): Attribut ou opération visible seulement à l'intérieur de la classe et pour toutes les sous-classes de la classe.
  - Privé (-) : Attribut ou opération seulement visible à l'intérieur de la classe.

3.1.2 Relations entre classes

Multiplicités (ou cardinalités) Les multiplicités permettent de contraindre le nombre d'objets intervenant dans les instanciations des associations. On en place de chaque côté des associations.Une multiplicité d'un côté spécifie combien d'objets de la classe du côté considéré sont associés à un objet donné de la classe de l'autre côté. Les principales multiplicités normalisées sont « plusieurs » (*), « exactement n » (n), « au minimum n » (n..*) et « entre n et m » (n..m). Il est aussi possible d'avoir un ensemble d'intervalles convexes: n1..n2, n3..n4 ou n1..n2, n5, n6..*, etc.

Association Une association est une relation entre deux classes (association binaire) ou plus (association n-aire), qui décrit les connexions structurelles entre leurs instances. L'association signifie qu'une classe contiendra une référence (ou un pointeur) de l'objet de la classe associée sous la forme d'un attribut. Chaque extrémité de l'association indique le rôle de la classe dans la relation et précise le nombre d'objets de la classe qui interviennent dans l'association. Quand l'information circule dans un sens uniquement, le sens de la navigation est indiqué à l'une des extrémités.
Sur un diagramme, il est possible de créer plusieurs associations entre les mêmes classes (associations multiples) et de créer une association d'une classe avec elle-même (association réflexive)

Figure 4: Représentation d'une association.

Classe-association Une association peut apporter de nouvelles informations (attributs et méthodes) qui n'appartiennent à aucune des deux classes qu'elle relie et qui sont spécifiques à l'association. Ces nouvelles informations peuvent être représentées par une nouvelle classe attachée à l'association via un trait en pointillés. Une classe-association peut être reliée à d'autres classes d'un diagramme de classes.

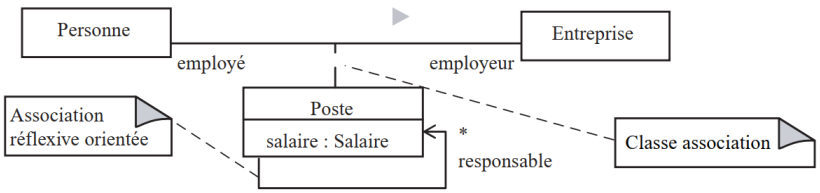

Figure 5: Représentation d'une classe-association et association réflexive

Agrégation Une agrégation est un cas particulier d'association non symétrique exprimant une relation de contenance. Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».
La destruction de l'agrégat n'entraîne pas la destruction de tous ses éléments.
Un élément peut appartenir à plusieurs agrégats (agrégation partagée).

Composition La composition est une relation d'agrégation **plus forte** dans laquelle il existe une contrainte de durée de vie entre la classe « composant » et la ou les classes « composé ». Autrement dit la suppression de la classe « composé » implique la suppression de la ou des classes « composant ».
Un élément ne peut appartenir qu'à un seul agrégat composite (agrégation non partagée).

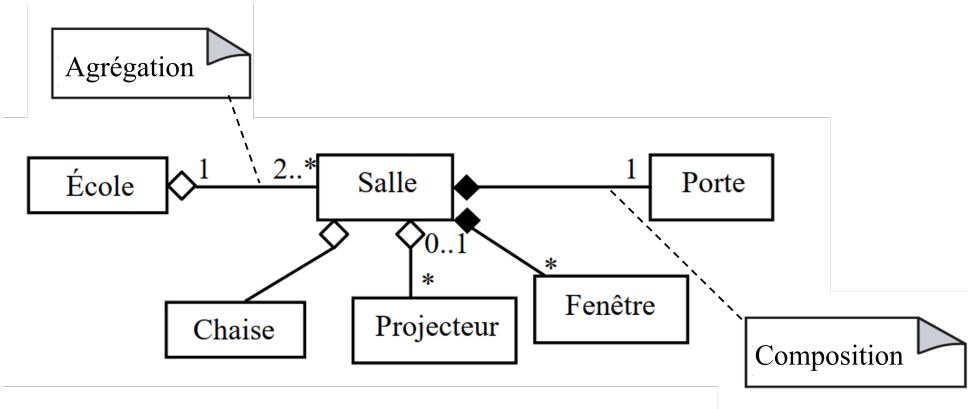

Figure 6: Agrégation et composition

Dépendance Une dépendance est une relation unidirectionnelle exprimant une dépendance sémantique entre des éléments du modèle. Elle indique que la modification de la cible peut impliquer une modification de la source.

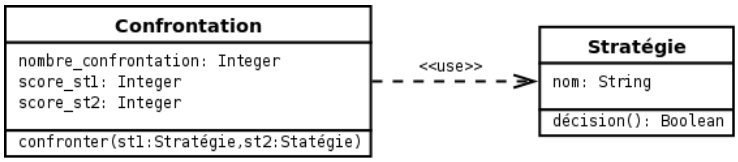

Figure 7: Représentation d'une dépendance.

Généralisation et Spécialisation La généralisation est la relation entre une classe et deux autres classes ou plus partageant un sous-ensemble commun d'attributs et/ou d'opérations. La classe qui est affinée s'appelle super-classe, les classes affinées s'appellent sous-classes. L'opération qui consiste à créer une super-classe à partir de classes s'appelle la généralisation. Inversement la spécialisation consiste à créer des sous-classes à partir d'une classe.

Classe Abstraite Une classe abstraite est simplement une classe qui ne s'instancie pas directement mais qui représente une pure abstraction afin de factoriser des propriétés. Lorsqu'une classe possède une seule spécialisation, alors elle ne doit pas être abstraite.

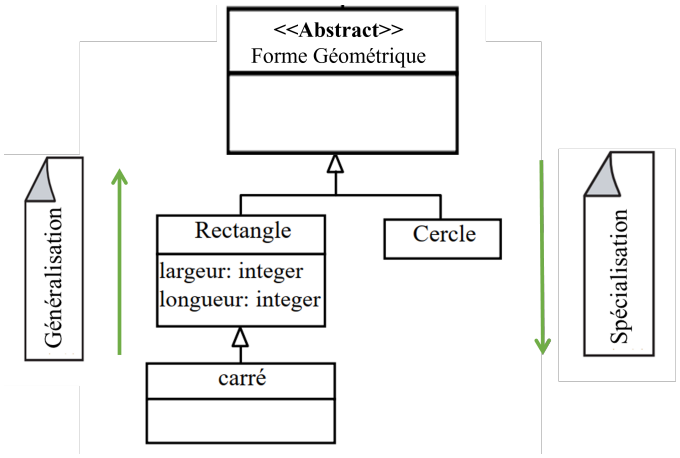

Figure 8: Représentation de l'héritage et classe abstraite.

Diagramme d'objet

Le diagramme d'objets représente les objets d'un système (c.a.d. les instances des classes) et leurs liens (c.a.d. les instances des associations) à un instant donné. Il donne une vue figée du système à un moment précis, par conséquent, à un diagramme de classe correspond une infinité de diagrammes d'objets. Ces derniers permettront d'affiner le diagramme de classe et de mieux le comprendre.

3.2 Classes de domaine

Le modèle de domaine définit les classes qui représentent les entités ou les concepts présents dans le domaine de l'application et les relations entre ces éléments. Ces classes peuvent être identifiées directement à partir de la connaissance du domaine ou par des entretiens avec des experts du domaine. Elles ne décrivent pas des d'objets logiciels mais correspondent plutôt à des concepts «réels» qui font abstraction à des éléments clairement identifiables dans le domaine.

Pour élaborer le modèle de domaine, il faut identifier:

- les entités ou les objets du domaine,
- Les attributs des classes de domaine,
- les relations entre les classes de domaine.

Pour ce faire, nous pouvons utiliser **l'analyse linguistique**. Il s'agit d'une technique simple qui analyse du texte afin de repérer les noms et les groupes nominaux dans la description textuelle d'un domaine et de les considérer comme des classes ou attributs potentiels et de repérer les Verbes et phrases verbales et les considérer comme des relations.

Le point faible de cette méthode est l'imprécision du langage naturel :

- Différentes classes peuvent représenter le même concepts,
- Difficultés avec les termes difficiles, les synonymes et les homonymes.
- Difficulté à trouver les classes « cachées ».

Remarque

L'erreur la plus courante lors de la création d'un modèle du domaine consiste à modéliser un concept par un attribut alors que ce dernier devait être modélisé par une classe. Si la seule chose que recouvre un concept est sa valeur, il s'agit simplement d'un attribut. Par contre, si un concept recouvre un ensemble d'informations, alors il s'agit plutôt d'une classe qui possède elle-même plusieurs attributs.

Exemple

Nom du cas	Admettre un patient
Type	Principal
Acteurs	Secrétaire
Objectif	Permet à la secrétaire d'attribuer une chambre à un patient.
Pré-condition	La secrétaire doit être authentifiée
Scénario nominal	1) La secrétaire clique sur le bouton « Admettre un patient » 2) Le système lui affiche un formulaire contenant les renseignements sur l'hospitalisation du patient (dates entrée et sortie, service, etc.). 3) La secrétaire remplit le formulaire. 4) Le système valide le formulaire. 5) Le système affiche une liste des chambres libres dans ce service. 6) La secrétaire choisit une chambre. 7) Le système lui demande si elle veut attribuer la chambre a un patient existant ou de créer un nouveau patient. 8) La secrétaire choisit d'attribuer la chambre un patient existant. 9) Le système fait appel au cas "**mettre à jour un patient** ". 10) Le système enregistre les informations de l'hospitalisation du patient.

Figure 9: Analyse linguistique la fiche descriptive du cas d'utilisation *Admettre un patient.*

Mots	Élément correspondant
Patient	Classe
Secrétaire	Classe
Chambre	Classe
Attribuer une chambre a un patient	Association entre chambre et patient
Système	Ce n'est pas une classe
Renseignement sur l'hospitalisation du patient	Classe association
date d'entrée et sortie, service,etc	Attribut de la classe Hospitalisation
Service	Classe
Liste des chambres dans ce service	Agrégation
Chambres libres	Attribut de la classe Chambre

Table 1: Extraction des classes de domaine a partir de la fiche descriptive du cas d'utilisation *Admettre un patient*

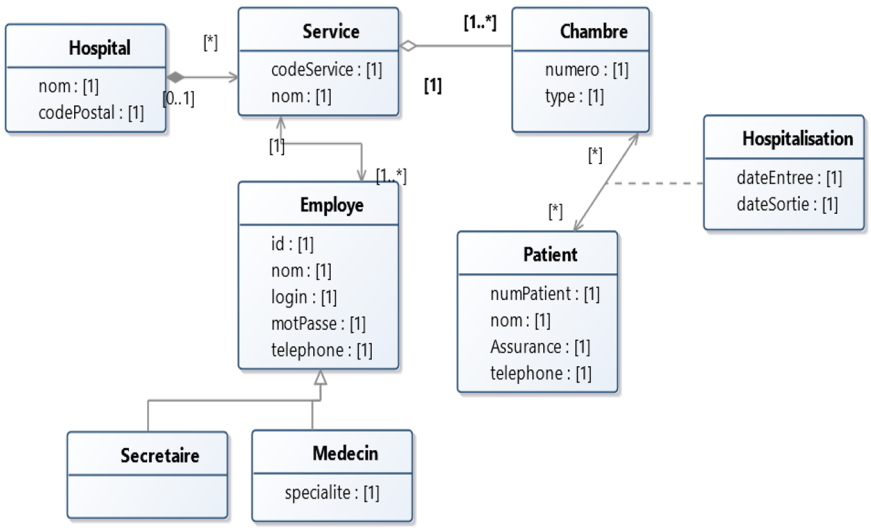

Figure 10: Diagramme de classes de domaine.

3.3 Classes d'analyse

3.3.1 Modèle MVC

Le MVC (Modèle-Vue-Contrôleur) est un modèle permettant de diviser en modules logiques une application comportant des interfaces graphiques, des données et de la logique du contrôle. Il impose la séparation entre les données, les traitements et la présentation, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface.

- Modèle : un noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.
- Vue : composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur.
- Contrôleur : composant responsable des prises de décision, gère la logique du code qui prend des décisions, il est l'intermédiaire entre le modèle et la vue.

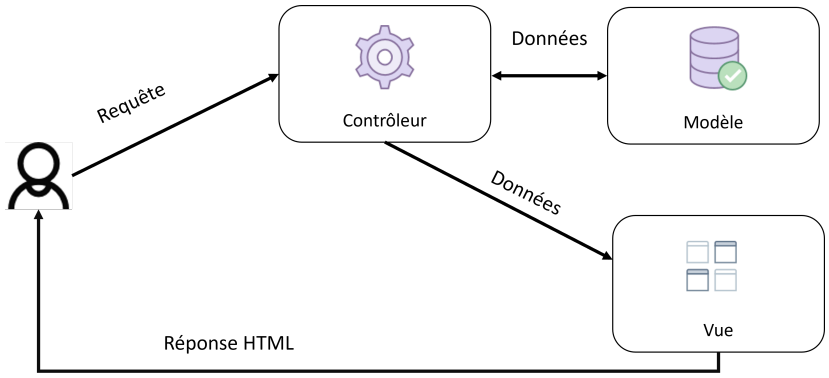

Figure 11: Le modèle MVC.

3.3.2 Diagramme de classes participantes

Les stéréotypes de Jacobson est une technique complémentaire de l'analyse linguistique. Cette technique permet de trouver les classes participantes dans la réalisation d'un cas d'utilisation donné en respectant le modèle MVC (Model-View-Control). Un avantage important de cette technique pour le chef de projet consiste en la possibilité de découper le travail de son équipe d'analystes suivant les différents cas d'utilisation, plutôt que de vouloir tout traiter d'un bloc. Les diagrammes de classes participantes sont donc particulièrement importants car ils font la jonction entre les cas d'utilisation, la maquette et le modèle de domaine.

Remarque

Un stéréotype est un mécanisme d'extensibilité permettant de créer de nouveaux éléments dérivés de ceux existants mais qui sont adaptés à des usages spécialisés dans des domaines particuliers.

Un diagramme de classes participantes est composé de:

- **Classes Dialogues** permettent les interactions entre le système et ses utilisateurs. Elles représentent les fenêtres et pages Web proposés à l'utilisateur : les formulaires de saisie, les résultats de recherche, etc. Elles proviennent directement de l'analyse de la maquette établis dans l'activité précédente. Elle vivent seulement le temps durant lequel le cas d'utilisation est concerné
- **Classes Contrôles** font la transition entre les dialogues et les concepts du domaine en leur permettant de manipuler des informations détenues par des objets métier. Elles visent la séparation des objets d'interface des données persistantes pour focaliser sur les fonctionnalités ou le comportements requis par les cas d'utilisation.
- **Classes Entités** représentent les concepts du domaine. Leur durée de vie est plus longue que celle des cas d'utilisation. Elles représentent généralement les données persistantes (permettent à des données ou des relations d'être stockées dans des fichiers ou des bases de données).

Exemple

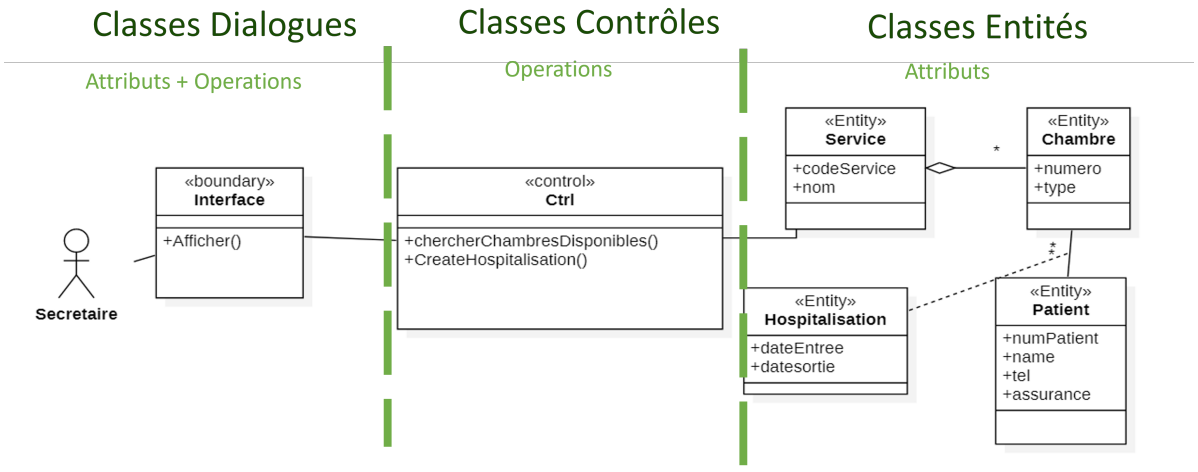

Figure 12: Diagramme de classes participantes du cas d'utilisation *Admettre un patient.*

4 Modèle d'analyse dynamique

4.1 Diagramme de Séquence détaillé

Une fois le modèle d'analyse objet est établi et les classes d'analyse sont définies, les analystes procèdent a la répartition du tout le comportement du système entre les classes d'analyse c.a.d l'attribution des responsabilités bien précises aux classes d'analyse et déterminer les services ou opérations réalisées par chaque classe.
Les diagrammes de séquence facilitent l'allocation de responsabilités en décrivant les interactions entre les objets du système. Par conséquent, le système n'est plus vu comme une boîte noire mais comme un ensemble d'objets collaborant dans le cadre d'un scénario d'exécution du système.
Un diagramme de séquence détaillé est un raffinement d'un diagramme de séquence système d'un cas d'utilisation donne. Il décrit généralement la création et destruction des instances des classes participantes ainsi que les messages échangés entre eux.

Création et destruction des objets Si un objet est créé par une opération, celui-ci n'apparaît qu'au moment où il est créé. Si l'objet est détruit par une opération, la destruction se représente par '*X*'.

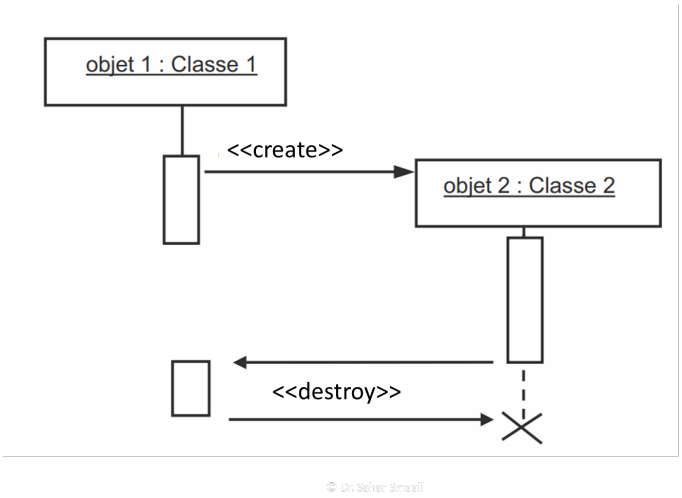

Figure 13: Création et destruction des objets.

Règles du modèle MVC Le diagramme de séquence détaillé utilise les stéréotypes de Jacobson et respectent les règles du modèle MVC suivantes:

- Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.
- Les dialogues peuvent interagir avec les contrôles.
- Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.
- Les entités ne peuvent interagir qu'entre elles.

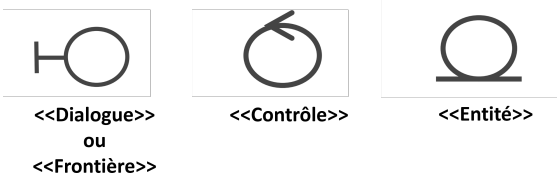

Figure 14: Représentation des objets dans un diagramme de séquence détaillé

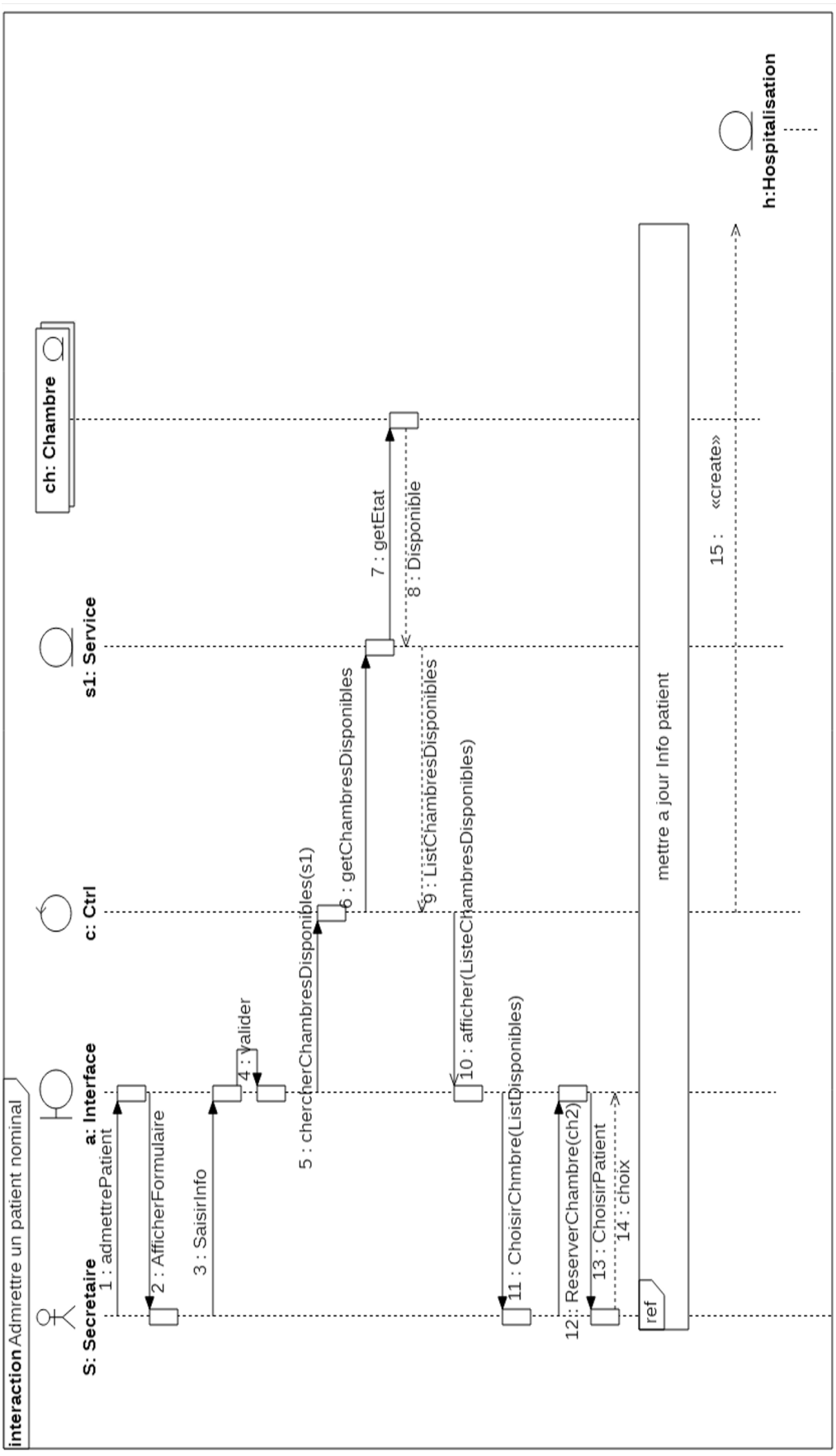

Figure 15: Diagramme de séquence détaillé du cas d'utilisation *Admettre un patient.*

Exemple

Utilité des diagrammes de séquence détaillés

- Ajouter ou préciser les opérations dans les classes (un message ne peut être reçu par un objet que si sa classe a déclaré l'opération publique correspondante).
- Ajouter des types aux attributs et aux paramètres et retours des opérations.
- Affiner les relations entre classes : associations, généralisations ou dépendances.

4.2 Diagramme d'états/transitions

Les diagrammes de séquence représentent le comportement du système du point de vue d'un cas d'utilisation unique. Cependant, un diagramme d'états/transitions décrive le comportement interne d'un seul objet indépendamment de son environnement à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode).

État d'un objet L'état d'un objet est défini, à un instant donné, par l'ensemble des valeurs de ses propriétés. Il représente une situation durant la vie d'un objet pendant laquelle :

- il satisfait une certaine condition,
- il exécute une certaine activité,
- ou bien il attend un certain événement.

Un objet passe par une succession d'états durant son existence. Un état a une durée finie, variable selon la vie de l'objet, en particulier en fonction des événements qui lui arrivent. L'ensemble des états du cycle de vie d'un objet contient :

- un état initial correspondant à l'état de l'objet juste après sa création (défini par l'un des constructeurs de l'objet).
- un ou plusieurs états finaux correspondant à une phase de destruction de l'objet.
  Il arrive également qu'il n'y ait pas d'état final car un objet peut ne jamais être détruit (dans le cas des services notamment).

Évènement est un fait qui déclenche le changement d'état, qui fait donc passer un objet d'un état à un autre état (désactivation d'un état et activation de l'état suivant). Il existe quatre types d'événements :

- **call** : généré par un appel de méthode sur l'objet courant.
- **signal** : généré par la réception d'un signal asynchrone, explicitement émis par un autre objet,Exemple : clic de souris, et interruption d'entrées-sorties.
- **Change**: C'est le cas de l'évaluation d'une expression booléenne (passage de faux a vrai).
- **after** et **when** : généré par l'écoulement d'une durée déterminée après un événement donné

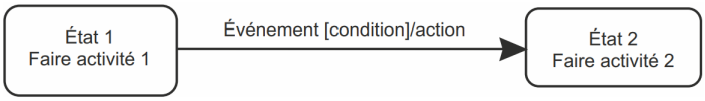

Figure 16: Diagramme d'états/transitions

Transition Une transition définit la réponse d'un objet à l'arrivée d'un événement. Elle indique qu'un objet qui se trouve dans un état peut « transiter » vers un autre état en exécutant éventuellement certaines activités, si un événement déclencheur se produit et qu'une condition de garde est vérifiée.

Action et activité Une action est une opération instantanée qui ne peut être interrompue ; elle est associée à une transition. Une activité est une opération d'une certaine durée qui peut être interrompue, elle est associée à un état d'un objet.

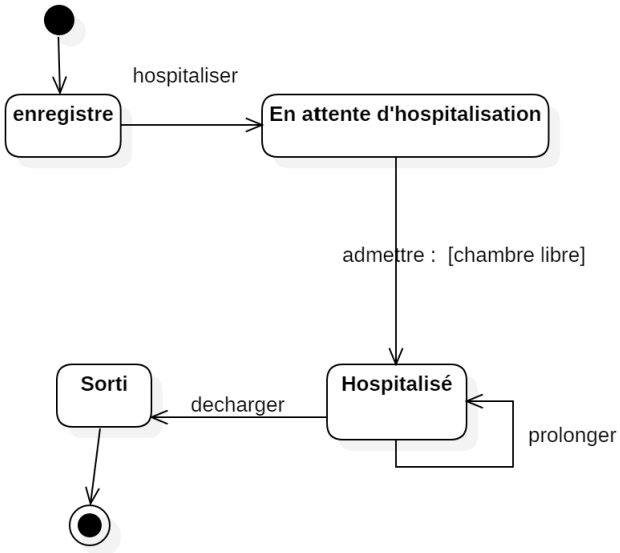

Figure 17: Diagramme états/transitions d'une entité patient.

Exemple

Références

- GABAY, Joseph et GABAY, David. UML 2 Analyse et conception: Mise en œuvre guidée avec études de cas. Dunod, 2008.
- ARLOW, Jim et NEUSTADT, Ila. UML 2 and the unified process: practical object-oriented analysis and design. Pearson Education, 2005.
- Pascal Roques. Les cahiers du programmeur, UML2 Modéliser une application web. 4eme édition. EYROLLES, 2008.
- UP : Unified Process, ÉDITIONS EYROLLES https://sabricole.developpez.com/uml/tutoriel/unifiedProcess/LIV, consulté le 29-11-2020.
- LETHBRIDGE, Timothy C. et LAGANIÈRE, Robert. Object-Oriented Software Engineering: Practical Software Development using UML and Java, 2004.
- Benoît Charroux,Aomar Osmani, Yann Thierry-Mieg, UML2 Pratique de la modélisation, 2e édition, Pearson Education, France 2009.