République du Cameroun
Paix –Travail – Patrie

Université de Yaoundé I
Sapienta – Collativia – Cognitio

Ecole Nationale Supérieure
Polytechnique de Yaoundé

Département de génie Informatique

Republic of Cameroon
Peace –Work – Fatherland

University of Yaoundé I
Sapienta – Collativia – Cognitio

National Advanced School of
Engineering of Yaounde

Department of Computer Engineering

SYSTÈME MULTIAGENT DE GESTION D'UN CABINET MEDICAL

NOMS ET PRENOMS DES MEMBRES DU GROUPE 7 :

ALETANOU LOIC	20P082
DIFFO TAHBOH MANUELLA	22P140
DEUTCHOUA MHOH IVAN PARFAIT	21P490
DONFACK KEUBOU AUDREY HILARY	20P189
KEUMOUO TADAHA DIROIL	20P172
MBOUMELA SOB ELTON LEWIS	20P522
MFANGAM NDASSA YASMINE AMIRAH	20P283
TCHATCHUENG DJOUM GILLES BORIS	20P221
TENE FOGANG ZACHARIE IGOR	20P540

Sous la coordination de : Pr. BATCHAKUI BERNABE

ANNEE ACADEMIQUE:
2023/2024

Table Des Matières

1 INTRODUCTION 4
2 PRESENTATION GENERALE DU PROJET 5
2.1 Situation 5
2.2 Contexte 5
2.3 Objectif principal 5
3 SPÉCIFICATION DES BESOINS 6
3.1 Les besoins fonctionnels 6
3.2 Besoins non fonctionnels 6
4 ANALYSE ET CONCEPTION 7
4.1 Architecture du système 7
4.2 Diagramme de contexte 8
4.3 Diagramme de cas d'utilisation 9
4.3.1 Diagramme 9
4.3.2 Description des cas d'utilisation 10
4.4 Diagramme de séquence 12
4.4.1 Prendre rendez-vous 12
4.4.2 Rediriger patient 13
4.4.3 Consulter patient 14
4.5 Diagramme d'activité général 15
4.6 Diagramme d'interactions 16
5 DÉVELOPPEMENT DU SYSTÈME MULTI-AGENT 17
5.1 Description des agents 17
5.1.1 Agent Patient (Agent central) 17
5.1.2 Agent Secrétaire 17
5.1.3 Agent Médecin 18
5.2 Modélisation du parcours du patient 20
5.3 Les outils technologiques 21
5.3.1 Le langage de programmation : Java 21
5.3.2 Plateforme de développement : JADE 21
6 CONCLUSION 22
7 RÉFÉRENCES BIBLIOGRAPHIQUES 23
8 ANNEXE 24

Table Des Figures :

Figure 1 Architecture générale du système 7
Figure 2 Diagramme de contexte 8
Figure 3 Diagramme de cas d'utilisation 9
Figure 4 Cas d'utilisation prendre rendez-vous 12
Figure 5 Cas d'utilisation Rediriger patient 13
Figure 6 Cas d'utilisation Consulter patient 14
Figure 7 Diagramme d'activité general 15
Figure 8 Diagramme d'interaction 16
Figure 9 Parcours du patient 20

1 INTRODUCTION

Le système multi-agent pour un cabinet médical représente une réponse innovante aux défis rencontrés dans la gestion des ressources et la prestation des soins de santé. Dans un contexte médical en constante évolution, où la coordination efficace des activités et la gestion des informations sont cruciales, l'intégration de technologies avancées telles que les systèmes multi-agents offre des perspectives prometteuses pour améliorer l'efficacité opérationnelle et la qualité des services médicaux. Ce projet vise à développer un système informatique capable de coordonner les différentes activités au sein d'un cabinet médical, en automatisant les processus de planification, de suivi des patients et de communication entre les professionnels de la santé. L'objectif principal est de concevoir une solution technologique qui optimise la gestion des ressources et facilite la prise de décision, afin d'offrir des soins de santé plus efficaces et personnalisés. Pour atteindre cet objectif, notre travail sera structuré en plusieurs phases clés, commençant par la spécification des besoins fonctionnels et non fonctionnels du système. Ensuite, une analyse approfondie sera menée pour concevoir l'architecture du système et développer les différents diagrammes de modélisation nécessaires à sa compréhension et sa mise en œuvre. Enfin, le développement du système multi-agent sera réalisé en suivant une approche itérative et incrémentale, permettant d'adapter le système aux besoins spécifiques du cabinet médical.

2 PRESENTATION GENERALE DU PROJET

2.1 SITUATION

Le secteur hospitalier est en pleine transformation. Les administrateurs des établissements de santé doivent gérer des ressources hospitalières limitées nécessitant des décisions qui peuvent s'avérer importantes. De plus, ils doivent faire face à un accroissement continuel des exigences des patients, des demandes de soins, de l'évolution des pathologies ainsi qu'à l'explosion des coûts de soins. Il est alors inévitable de recourir à des améliorations des activités de la logistique hospitalière pour une meilleure gestion.[1]

2.2 CONTEXTE

Le système de santé actuel est confronté à des défis majeurs, tels que l'augmentation de la demande en soins, la pénurie de ressources et la nécessité de réduire les coûts. Ainsi, les cabinets médicaux jouent un rôle crucial en offrant des soins de proximité aux patients. Cependant, la gestion efficace d'un cabinet médical est une tâche complexe qui nécessite une coordination optimale des ressources et une communication fluide entre les différents acteurs. Les systèmes multi-agents (SMA) font partie des technologies prometteuses qui peuvent contribuer à la transformation du système de santé. Notre projet propose une solution innovante basée sur l'utilisation de systèmes multi-agents (SMA) pour optimiser la gestion des cabinets médicaux. Les SMA sont des systèmes informatiques composés d'entités autonomes appelées "agents" qui peuvent interagir entre eux et avec leur environnement de manière flexible et adaptative.

2.3 OBJECTIF PRINCIPAL

L'objectif principal de notre projet de gestion d'un cabinet médical avec des systèmes multi agents est d'améliorer l'efficacité et la qualité des soins fournis aux patients grâce à l'utilisation de technologies avancées de gestion et de collaboration. Il vise à intégrer des systèmes multi agents pour automatiser les tâches répétitives, faciliter la prise de décision et permettre une collaboration intelligente entre les différentes entités du cabinet médical.

3 SPÉCIFICATION DES BESOINS

3.1 LES BESOINS FONCTIONNELS

- Le système devra gérer l'arrivée et la prise en charge d'un patient

- Le système devra gérer les rendez-vous : Permettre à la secrétaire de planifier, modifier et annuler des rendez-vous pour les patients en fonction des disponibilités du médecin.

- Le système devra gérer la communication entre les Acteurs : Faciliter la communication entre la secrétaire, le patient et le médecin pour des questions liées aux rendez-vous, aux traitements, aux résultats d'examens, etc.

- Le système devra permettre la consultation d'un patient.

3.2 BESOINS NON FONCTIONNELS

- **Performance** : Le système doit être capable de gérer un grand nombre d'utilisateurs et de requêtes simultanément.

- **Sécurité** : Le système doit protéger les données médicales des patients contre l'accès non autorisé. Le système doit être conforme aux réglementations en matière de confidentialité des données.

- **Scalabilité** : Le système doit pouvoir être facilement évolutif pour répondre à l'augmentation du nombre d'utilisateurs et de données. Le système doit pouvoir être intégré à d'autres systèmes de santé.

- **Utilisabilité** : Le système doit être facile à utiliser

- **Maintenabilité** : Le système doit être conçu pour être évolutif.

4 ANALYSE ET CONCEPTION

4.1 ARCHITECTURE DU SYSTÈME

La figure 1 nous présente l'architecture de notre système multi agent de cabinet médical:

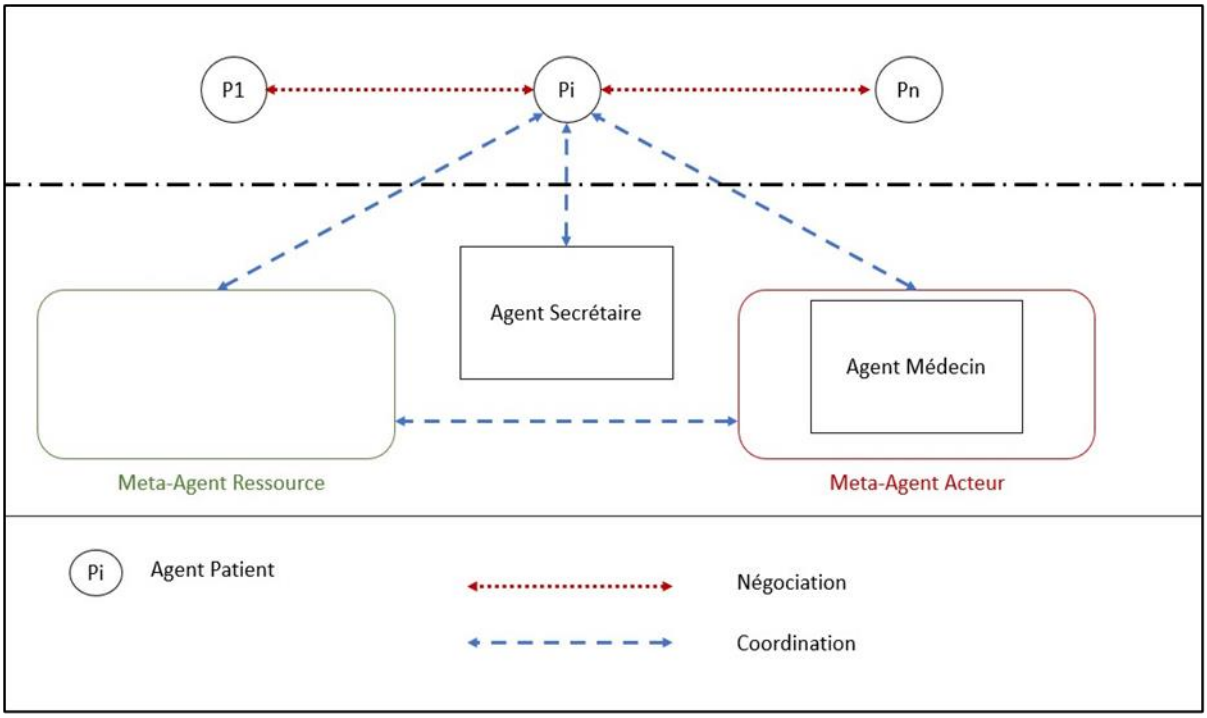

Figure 1 Architecture générale du système

Nous avons opté pour une architecture hétérogène et hiérarchique basée sur les agents en plaçant autour de l'agent central, qui est « l'Agent Patient », des agents auxiliaires qui représentent les ressources partagées du cabinet médical. Tout d'abord, cette approche permet une flexibilité accrue en termes de contrôle, car différents niveaux de décision peuvent être pris en fonction de la nature spécifique de chaque activité à coordonner. Ainsi, certains conflits ou problèmes peuvent être résolus de manière autonome à un niveau inférieur, tandis que d'autres nécessitent une coordination ou une négociation entre plusieurs entités au niveau supérieur. Deuxièmement, cette architecture permet une variété de modes de communication, notamment la coordination, la coopération et la négociation, en fonction des besoins et des exigences de chaque situation. Enfin, la structure hétérogène offre une grande adaptabilité, car de nouvelles entités peuvent être ajoutées ou supprimées sans nécessiter de modifications majeures dans la conception globale du système.

En résumé, cette approche favorise la résilience, la diversité des interactions et la capacité d'évolution du système dans un environnement dynamique.

4.2 DIAGRAMME DE CONTEXTE

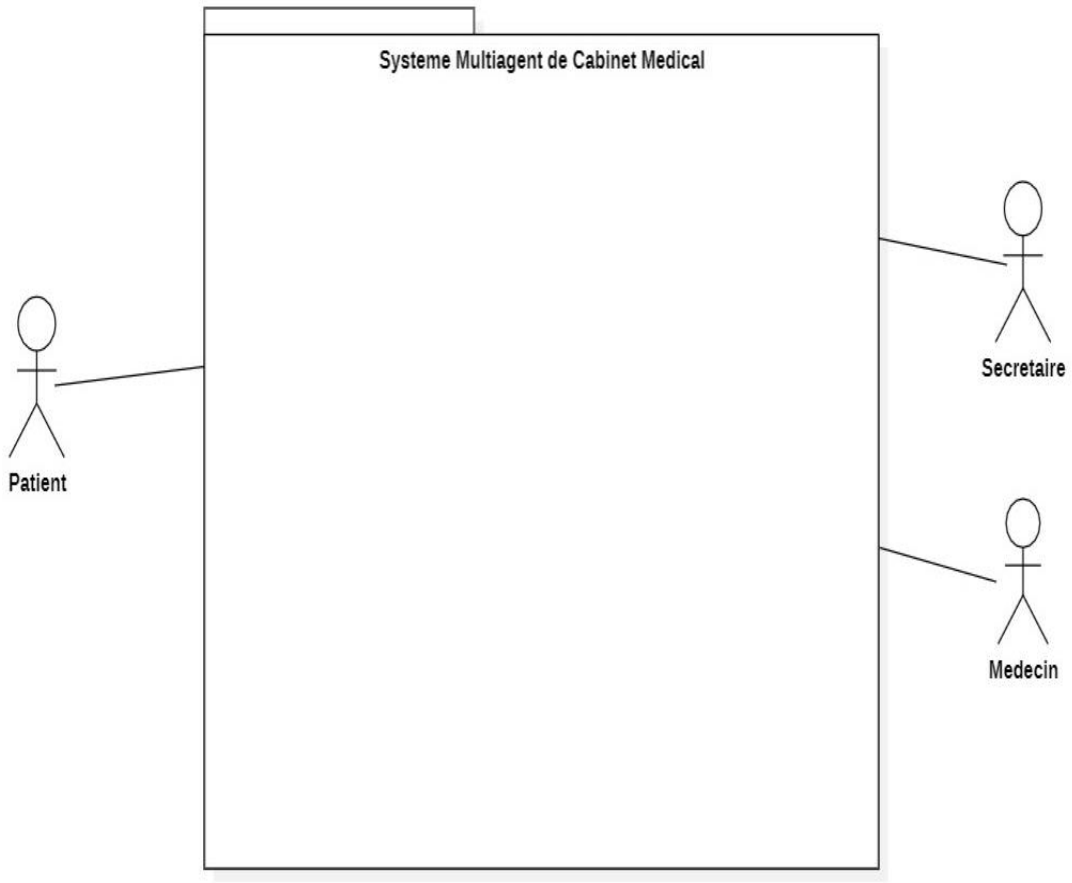

Figure 2 Diagramme de contexte

4.3 DIAGRAMME DE CAS D'UTILISATION

4.3.1 Diagramme

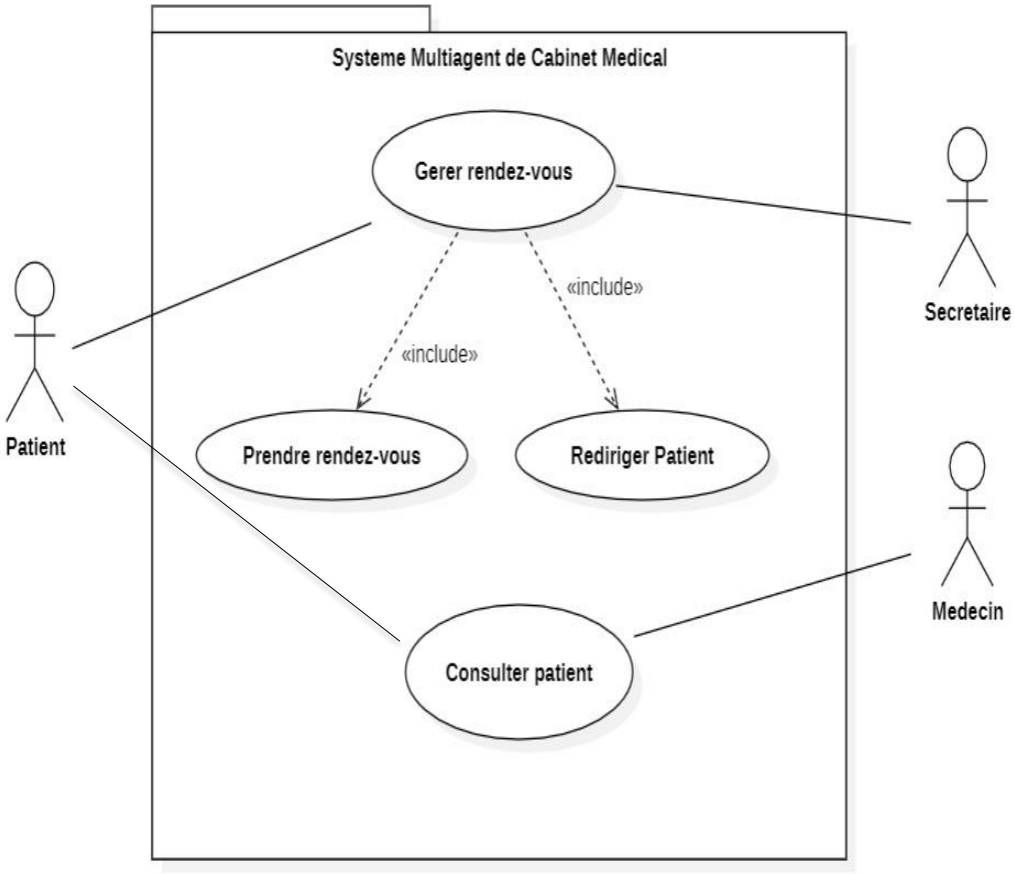

Figure 3 Diagramme de cas d'utilisation

4.3.2 Description des cas d'utilisation

Cas d'utilisation	**Gérer rendez-vous**
Acteur	Secrétaire, Patient
Description	Permet la gestion des rendez-vous médicaux entre les patients et les médecins.
Post Condition	Le rendez-vous est pris, modifié ou annulé selon les demandes des utilisateurs, et les modifications sont enregistrées dans le système

Cas d'utilisation	**Prendre rendez-vous**
Acteur	Secrétaire, Patient
Description	Permet à un patient de prendre un rendez-vous avec un médecin via le secrétaire.
Post Condition	Le rendez-vous est pris avec succès et enregistré dans le système

Cas d’utilisation	**Rediriger patient**
Acteur	Secrétaire, Patient
Description	Permet à la secrétaire d'accompagner ou de guider un patient vers le cabinet du médecin pour son rendez-vous planifié
Post Condition	Le patient est accompagné avec succès jusqu'au cabinet du médecin pour son rendez-vous, assurant ainsi un démarrage efficace de la consultation médicale.

Cas d’utilisation	**Consulter patient**
Acteur	Médecin, Patient
Description	Permet au médecin de consulter un patient lors d'une visite médicale programmée
Post Condition	La consultation médicale est réalisée avec succès et les informations sont enregistrées dans le dossier médical électronique du patient

4.4 DIAGRAMME DE SÉQUENCE

4.4.1 Prendre rendez-vous

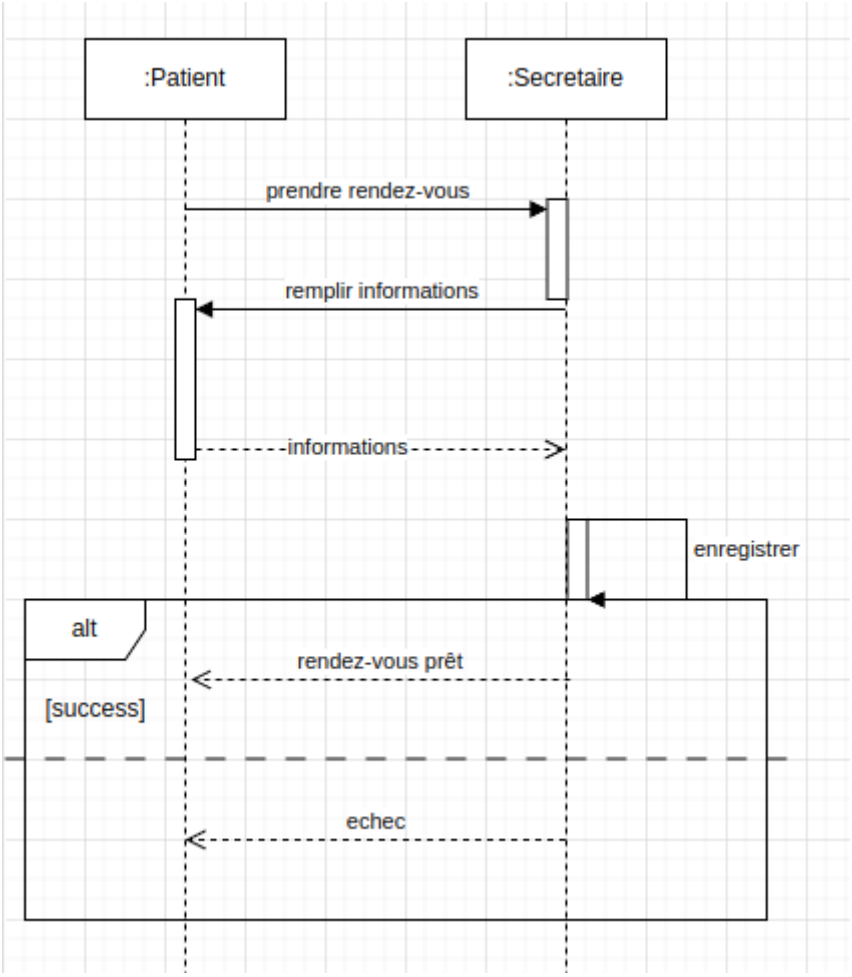

Figure 4 Cas d'utilisation prendre rendez-vous

4.4.2 Rediriger patient

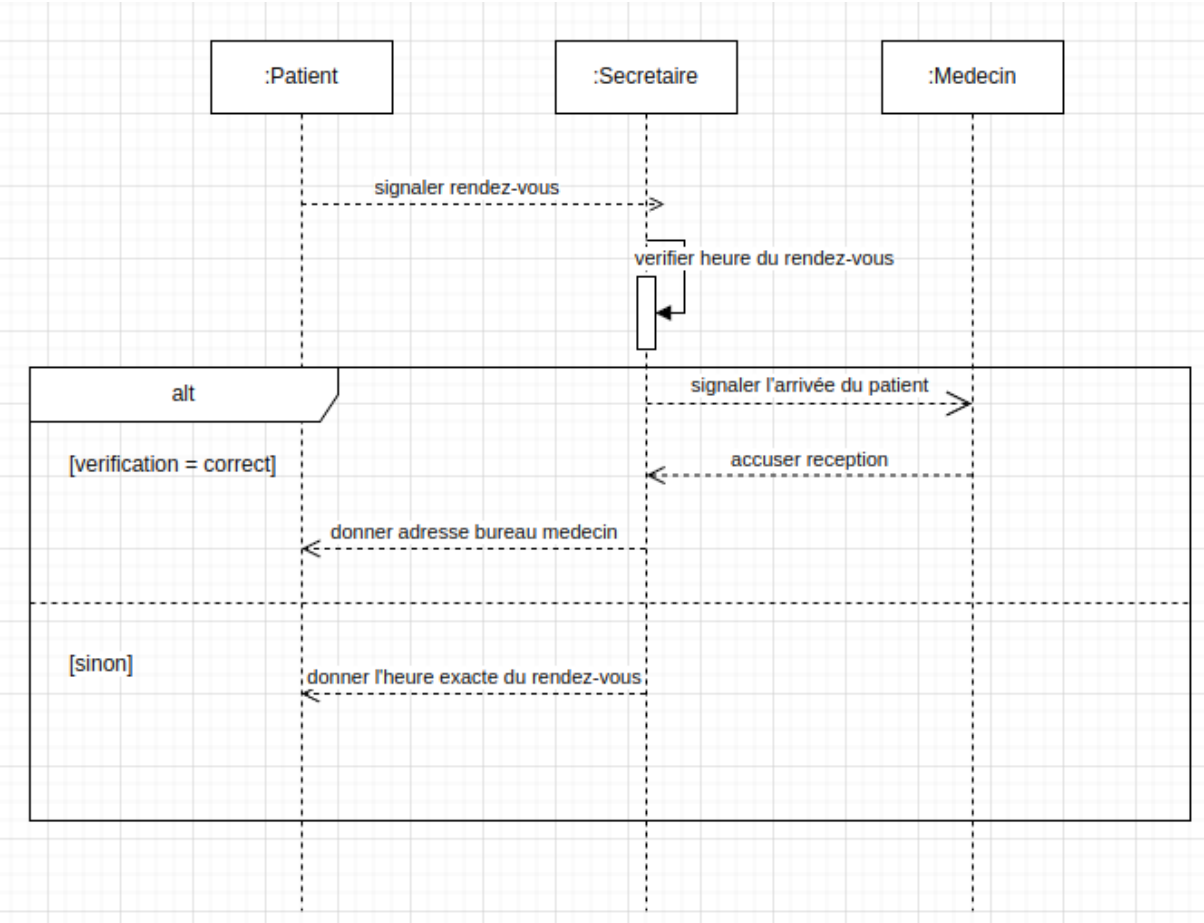

Figure 5 Cas d'utilisation Rediriger patient

4.4.3 Consulter patient

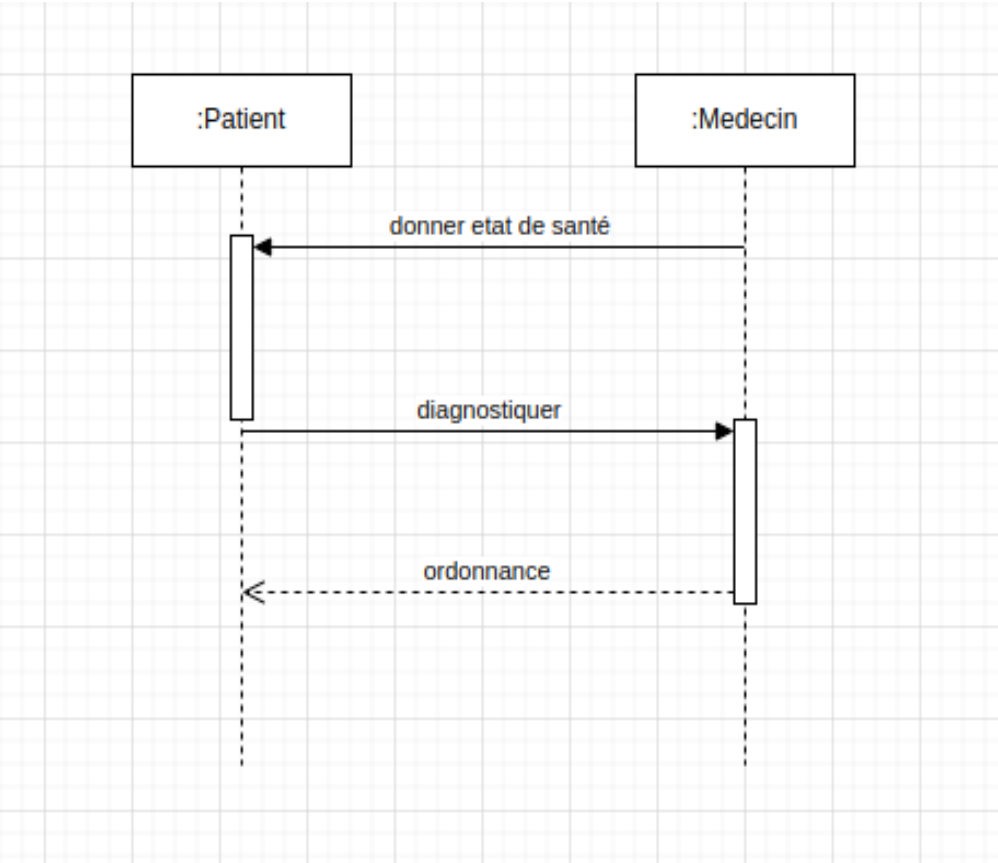

Figure 6 Cas d'utilisation Consulter patient

4.5 DIAGRAMME D'ACTIVITÉ GÉNÉRAL

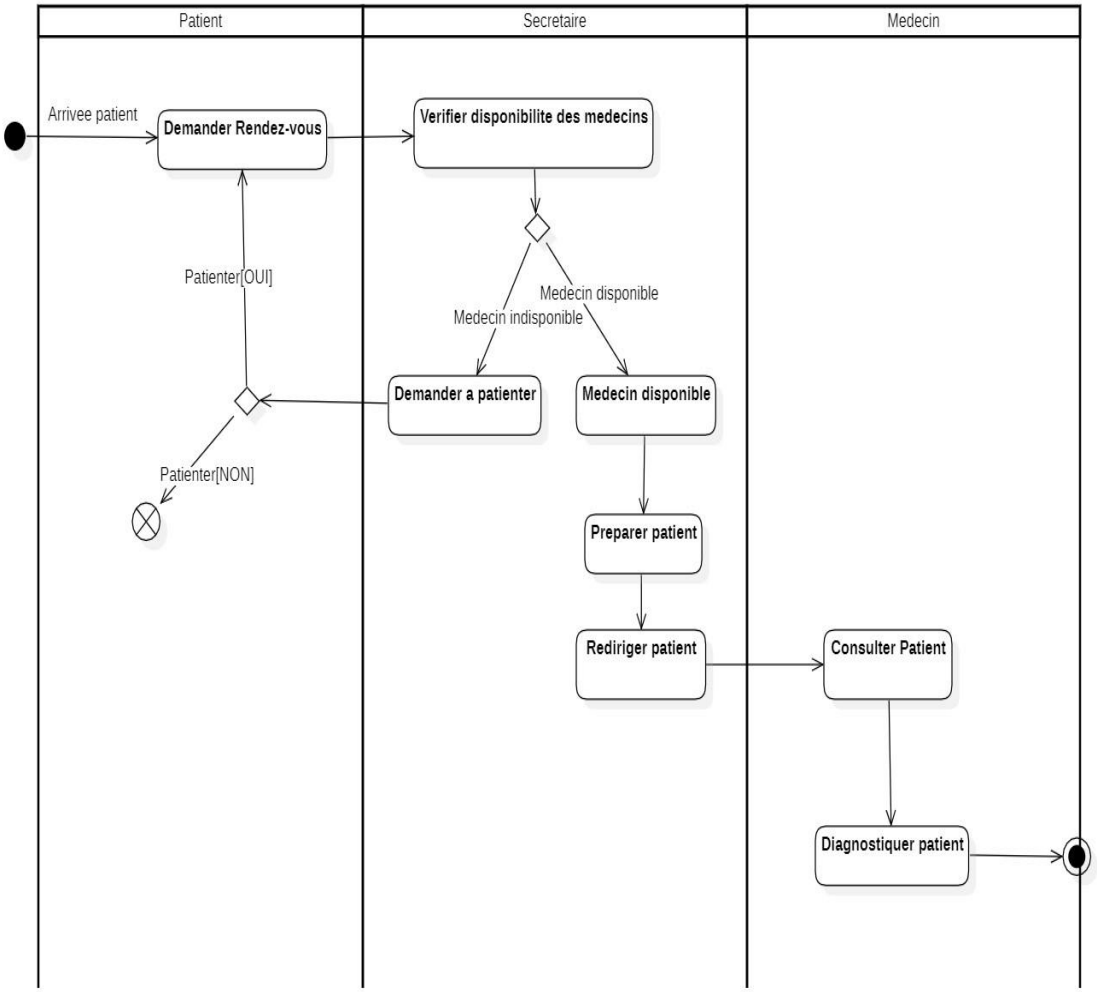

Figure 7 Diagramme d'activité général

4.6 DIAGRAMME D'INTERACTIONS

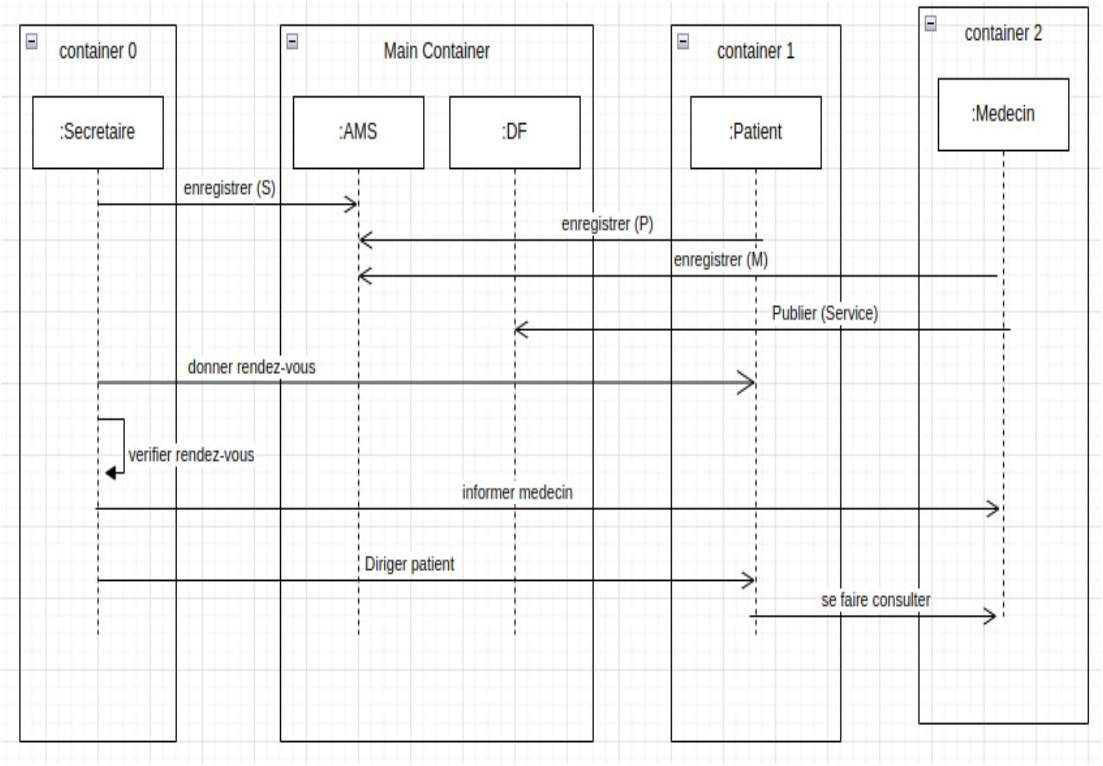

Figure 8 Diagramme d'interaction

5 DÉVELOPPEMENT DU SYSTÈME MULTI-AGENT

5.1 DESCRIPTION DES AGENTS

5.1.1 Agent Patient (Agent central)

Avant de poursuivre, il est crucial de clarifier la distinction entre "Agent Patient" et le patient. L'Agent Patient est une application informatique cognitive qui représente le patient au sein du système, tandis que le patient réel est la personne physique qui reçoit les soins.

Modélisation de l'Agent Patient

➢ **Perceptions**

- Son état de santé
- Les symptômes qu'il ressent
- Les rendez-vous pris chez le médecin
- Les résultats des analyses médicales

➢ **Influences** (actions)

- Prendre rendez-vous chez le médecin
- Annuler un rendez-vous
- Consulter le médecin
- Subir des analyses médicales

5.1.2 Agent Secrétaire

L'agent reçoit les messages entrants. Le message reçu est vérifié pour son ID de conversation afin de différencier les demandes de rendez-vous (« rendez-vous ») et les messages d'au revoir (« au revoir »). Si une demande de rendez-vous est reçue, l'agent vérifie la disponibilité des médecins. La méthode envoie des demandes de disponibilité aux médecins et attend leurs réponses.

- Si une demande de rendez-vous est reçue, l'agent vérifie la disponibilité des médecins Si un médecin est disponible, l'agent planifie le rendez-vous et informe

le patient. Si aucun médecin n'est disponible, l'agent informe le patient d'attendre ou de revenir plus tard.
- Si un message d'au revoir est reçu, l'agent envoie une réponse d'au revoir au patient.
- Si aucun message n'est reçu, le comportement se bloque à l'aide d'une méthode qui suspend le comportement jusqu'à ce qu'un nouveau message soit reçu.

Modélisation de l'Agent Secrétaire

➢ **Perceptions**

- Les rendez-vous pris par les patients
- La disponibilité du médecin
- Les informations ou identifiants des patients

➢ **Influences** (actions)

- Gérer les rendez-vous des patients (prise de rendez-vous, annulation, modification)
- Accueillir les patients
- Enregistrer les informations administratives des patients
- Préparer les dossiers médicaux des patients pour le médecin

5.1.3 Agent Médecin

L'Agent Secrétaire conduit le patient au médecin chargé de son suivi, si l'Agent Médecin correspondant n'est pas actif, le système le crée. Un deuxième exemple est quand le médecin élabore le plan de soins qu'il détaille dans son compte rendu. Ce dernier est envoyé à l'Agent Patient afin de planifier et ordonnancer ses activités de soins. L'Agent Médecin a un comportement prédictif. Il notifie l'Agent Patient d'éventuelles perturbations en cours de l'activité de soin. L'agent génère le planning de l'acteur correspondant et peut accéder aux dossiers médicaux des patients dont il assure le suivi.

Modélisation de l'Agent Médecin

➢ **Perceptions**

- Les symptômes du patient
- Les résultats des analyses médicales
- L'historique médical du patient

➢ **Influences** (actions)

- Consulter le patient
- Diagnostiquer la maladie du patient
- Prescrire un traitement au patient

5.2 MODÉLISATION DU PARCOURS DU PATIENT

Le flux des patients représente le flux principal. La notion de flux est associée à la notion de processus de prise en charge. Nous nous sommes intéressés à l'analyse et la modélisation du parcours des patients au niveau opérationnel. Cette cartographie du parcours simple des patients nous a permis de mieux comprendre le fonctionnement des systèmes de production des soins pour le cas d'un cabinet médical basique.

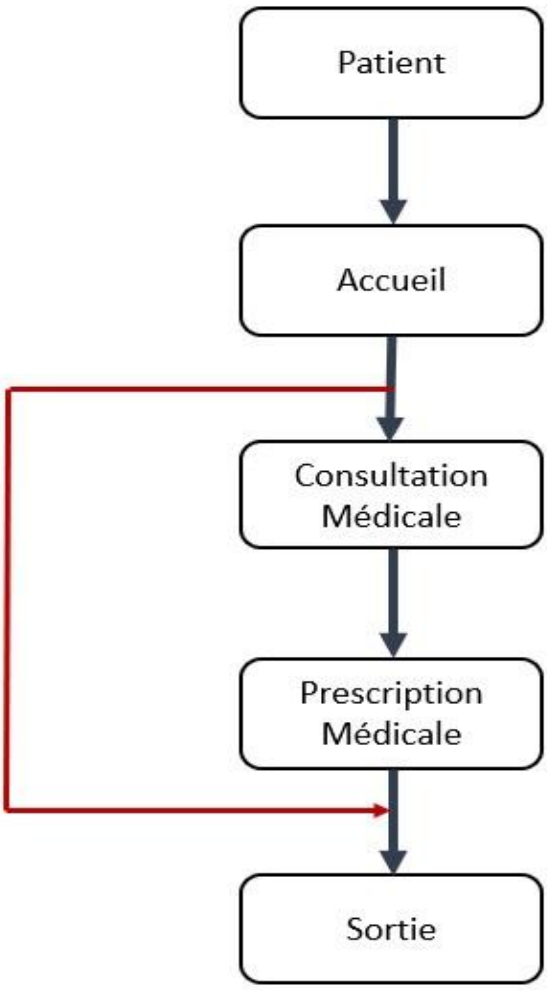

Figure 9 Parcours du patient

5.3 LES OUTILS TECHNOLOGIQUES

5.3.1 Le langage de programmation : Java

Java est un langage de programmation populaire et polyvalent. Il permet une programmation orientée-objet et modulaire. Outre son orientation objet, le langage Java est portable : un code Java est exécutable sur n'importe quelle machine (serveur, station de travail, ordinateur portable, tablette ou smartphone). Java permet également une exécution, en apparence, simultanée de plusieurs processus. De plus, de nombreuses bibliothèques de développement libres fournissent des mises en œuvre de plusieurs fonctionnalités facilement utilisables.

5.3.2 Plateforme de développement : JADE

Nous avons utilisé **JADE** (Java Agent Development Framework) pour la conception et l'implémentation de systèmes multi-agents. JADE est une plateforme multi-agents développée en Java par CSELT (groupe de recherche de Gruppo Telecom, Italie) pour simplifier la mise en œuvre des applications distribuées à base d'agents à travers un middleware. JADE offre une implémentation flexible des SMA qui communiquent via des messages ACL. De plus, cette plateforme supporte la mobilité et l'évolutivité. Elle offre également la possibilité d'intégration des services web, de distribution sur différents serveurs ainsi que la communication entre plusieurs plateformes JADE.

6 CONCLUSION

En conclusion, la conception d'un système multi-agent pour un cabinet médical représente une réponse innovante et prometteuse aux défis rencontrés dans la prestation des soins de santé. Notre projet a une opportunité unique d'améliorer l'efficacité opérationnelle, la coordination des activités et la qualité des services médicaux en exploitant les avancées technologiques en matière d'intelligence artificielle et de systèmes distribués. À travers notre étude, nous avons identifié les besoins spécifiques d'un cabinet médical en termes de gestion des ressources, de suivi des patients et de communication interprofessionnelle. La solution que nous avons proposée adopte une approche itérative et incrémentale pour le développement, ce projet assure la robustesse et la fiabilité du système, ouvrant ainsi la voie à une amélioration continue des soins de santé.

7 RÉFÉRENCES BIBLIOGRAPHIQUES

[1]- Noura Benhajji, Système multi-agents de pilotage réactif des parcours

patients au sein des systèmes hospitaliers, [en ligne], lien internet:https://www.mcours.net/cours/pdf/leilclic3/leilclic933.pdf , consulté le 16/04/2024.

[2]- Wikipedia, Système Multi agent , [en ligne], lien internet: https://fr.wikipedia.org/wiki/Syst%C3%A8me_multi-agents , consulté le 15/04/2024.

[3]- Jade site, Tutorials & Guides, [en ligne], lien internet: https://jade.tilab.com/documentation/tutorials-guides/ , consulté le 15/04/2024.

[4]- Mcours, Les Systèmes Multi-Agents, [en ligne], lien internet:https://www.mcours.net/cours/pdf/leilclic3/leilclic933.pdf , consulté le 15/04/2024.

8 ANNEXE

CREATION D'UN AGENT AVEC JADE

Jade est un middleware qui facilite le développement des systèmes multi agents (SMA).

JADE contient :

- Un runtime Environment : l'environnement ou les agents peuvent vivre. Ce runtime Environment doit être activé pour pouvoir lancer les agents ;
- Une librairie de classes : que les développeurs utilisent pour écrire leurs agents ;
- Une suite d'outils graphiques : qui facilitent la gestion et la supervision de la plateforme des agents.

I. Installation de jade

Voici les étapes à suivre pour installer JADE :

Téléchargez le fichier JADE-all-3.6.zip de l'adresse http://jade.tilab.com/download.php ;

Décompressez le fichier (on va supposer tout au long de ce tutoriel que le chemin du répertoire JADE-all-3.6 est le c:\ JADE-all-3.6). Après avoir décompressé le fichier, vous retrouvez quatre autres fichiers ZIP (JADE-bin-3.6.zip , JADE-doc-3.6.zip, JADE-examples-3.6.zip, JADE-src-3.6.zip). Décompressez ces quatre fichiers ;

On doit maintenant mettre à jour la variable classpath (si elle n'existe pas encore il faut la créer) en faisant comme suit :

- Sur le poste de travail, choisissez propriétés. La fenêtre propriétés système apparaît, choisissez l'onglet Avancé Puis cliquez sur variables d'environnement, Une petite fenêtre intitulée « variables d'environnement » apparaît.
- En suite sur la même fenêtre sélectionner la variable TEMP.
- Dans la zone variables système, essayez de trouver la variable d'environnement qui porte le nom CLASSPATH. Si vous ne la trouvez pas, si qu'elle n'existe pas il faudra la créer Maintenant que la variable est trouvée/créée on doit lui attribuer une valeur. Cette valeur est la concaténation des chemins des quatre fichiers jar http.jar, iiop.jar, jade.jar, jadeTools.jar situés dans le chemin c:\JADE-all-3.6\JADE-bin-3.6\jade\lib.

Vous devez obtenir quelque chose de similaire a :

C:\JADE-all-3.6\JADE-bin-3.6\jade\lib\http.jar;C:\JADE-all-3.6\JADE-bin-3.6\jade\lib\iiop.jar;

C:\JADE-all-3.6\JADE-bin-3.6\jade\lib\jade.jar;C:\JADE-all-3.6\JADE-bin-3.6\jade\lib\jadeTools.jar

Séparez les chemins par des ; et Sauvegardez les modifications. Pour vérifier que l'opération est bien réalisée, tapez dans la fenêtre exécuter(démarrer->Exécuter) ou dans l'invite de commande la commande suivante :

Java jade.Boot -gui

Une fenêtre s'ouvre et lance la plateforme jade.

II. Création d'un agent avec JADE

Créons un agent qui a le comportement d'une infirmière.

```
package firstAgent;

import jade.core.Agent;

public class NurseAgent extends Agent {

 protected void setup() {

    System.out.println("Bonjour, je suis une infirmière et je suis prête à aider les patients.");

    prendreConstantesVitales();

    gererRendezVous();

    gererDemandesPatients();

    // Autres méthodes de simulation de comportement d'une infirmière peuvent être appelées ici

 }

 private void prendreConstantesVitales() {

    System.out.println("Je prends les constantes vitales des patients.");

    // Code pour prendre les constantes vitales des patients

 }

 private void gererRendezVous() {

    System.out.println("Je gère les rendez-vous des patients.");
```

```
    // Code pour gérer les rendez-vous des patients
  }
  private void gererDemandesPatients() {
    System.out.println("Je réponds aux demandes des patients.");
    // Code pour répondre aux demandes des patients
  }
  // Autres méthodes de simulation de comportement d'une infirmière peuvent être ajoutées ici
}
```