École Polytechnique de Montréal

Baccalauréat en Génie Logiciel

Cours - LOG1000

Travail pratique 2

Auteurs :

Maxime Tremblay-Rheault - 1946878 Maude Desrosiers-Carbonneau - 1946801

7 octobre 2019

Le rapport porte sur les diagrammes de cas d'utilisation, de séquence et UML. Divers tableaux et figures seront présentés de manière succincte. Le patient et l'infirmière ont été définis comme étant les acteurs principaux.

La figure suivante représente les cas d'utilisations possible entre les acteurs principaux, soit le patient et l'infirmière.

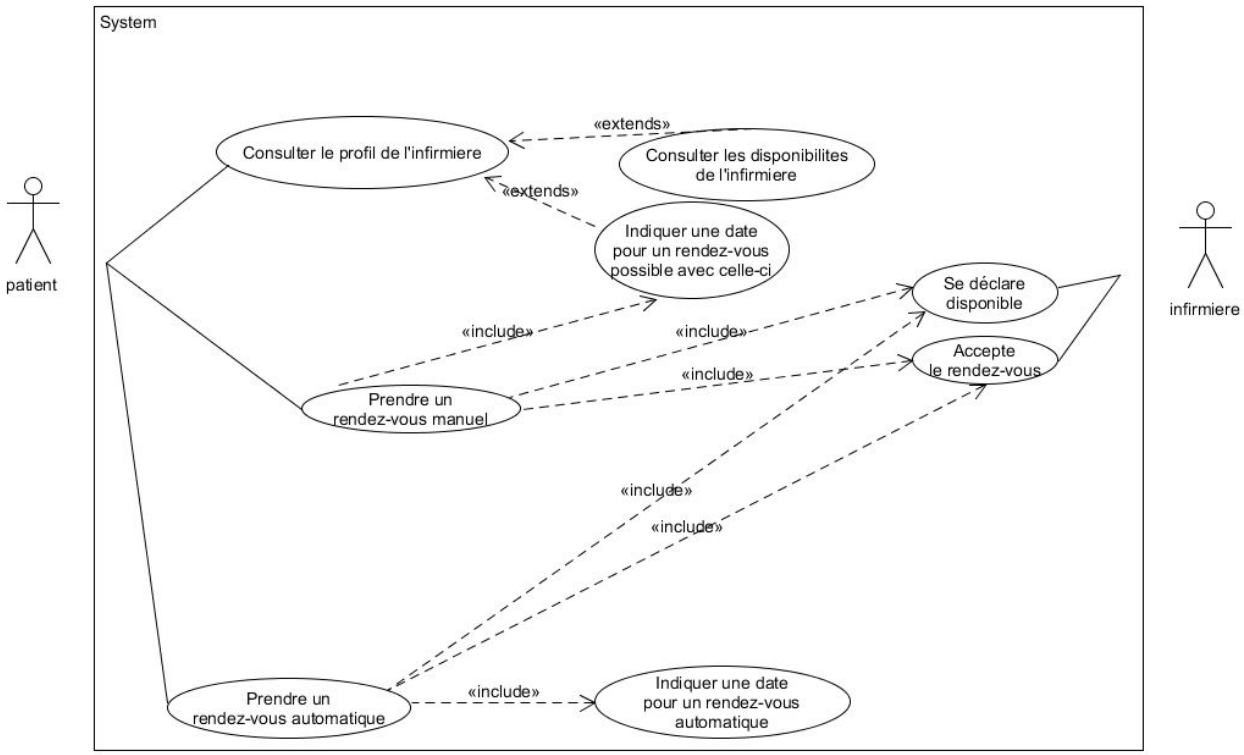


Figure 1 : Diagramme de cas d'utilisation

Le tableau suivant décrit textuellement un cas d'utilisation précis et définit des cas alternatifs si certaines conditions sont remplies.

Tableau 1 : Description d'un cas d"utilisation

Cas d'utilisation	**Prendre un rendez-vous automatique avec un infirmière**
Description	Le patient prend un rendez-vous automatique avec une infirmière après avoir indiqué une date
Précondition	L'utilisateur et l'infirmière possèdent un profil. L'utilisateur s'est authentifié.
Postconditions	En cas de succès: Le rendez-vous est ajouté au calendrier de l'hôpital En cas d'échec: Le système affiche un message d'erreur et offre au patient la possibilité de proposer une autre date ou de quitter le mode de rendez-vous automatique.
Conditions ...	Une infirmière se déclare disponible et accepte le rendez-vous
Acteurs impliqués	Patient et infirmière
Déclencheur	Action du patient
Flux actions principales	
1	Le patient décide de prendre un rendez-vous automatique
2	Le patient fournit une date et une heure pour le rendez-vous automatique
3	Le patient confirme la date pour le rendez-vous et l'exactitude de ses renseignements personnels
4	Le système vérifie la disponibilité des infirmières à la date fournie par le patient
5	Le système propose le rendez-vous à une infirmière disponible
6	L'infirmière accepte ou non le rendez-vous
7	Le rendez-vous est ajouté au calendrier de l'hôpital
8	Le système envoie une confirmation du rendez-vous avec le nom de l'infirmière qui a accepté le rendez-vous
Extension	
3.A	Les informations conviennent-elles?
3.A.1	Si oui, continuer
3.A. 2	Si non, offrir de revenir à l'étape 2 ou ou de quitter le mode de rendez-vous automatique pour pouvoir modifier ses informations personnelles.
4.A	Y a-t-il une infirmière disponible?
4.A.1	Si oui, aller à l'étape 5
4.A.2	Si non, avertir le patient que la date choisie ne convient pas et le ramener à l'étape 1.
6.A	L'infirmière accepte-t-elle le rendez-vous proposé?
6.A.1	Si oui, aller à l'étape 7
6.A.2	Si non, répéter l'étape 6 avec la prochaine infirmière disponible. Au cas où aucune infirmière accepte le rendez-vous, afficher un message d'avertissement au patient et le rediriger à l'étape 1 ou lui offrir la possibilité de quitter le mode de rendez-vous automatique.

On constate que la prise de rendez-vous implique huit opérations, dont plusieurs nécessitent des conditions préalables.

La figure suivante illustre un diagramme de séquence. Le diagramme permet de visualiser temporellement la séquence d'actions.

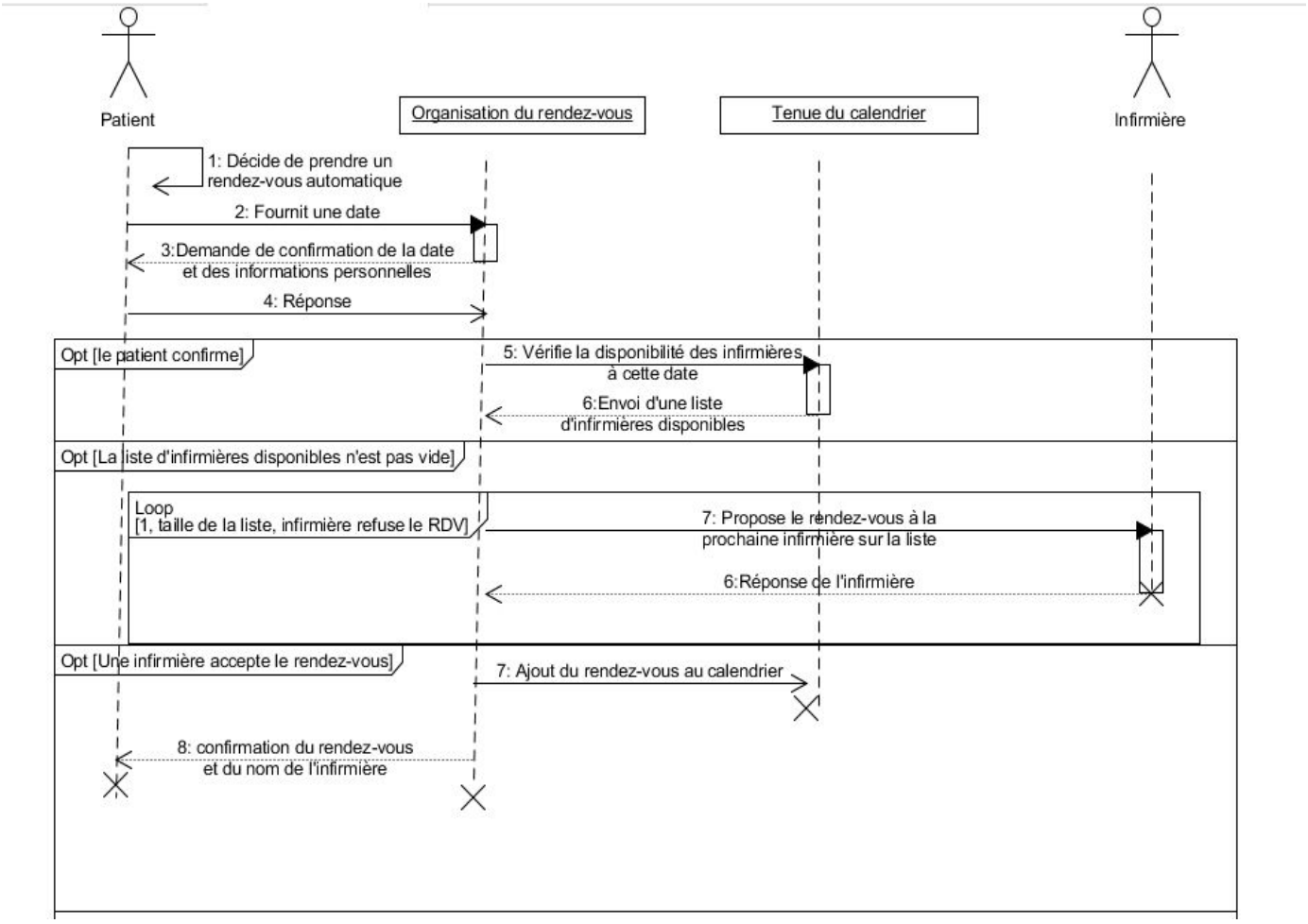


Figure 2 : Diagramme de séquence

Les rôles, titrés au-dessus de la figure, sont pris en charge par des instances de classes. Celles-ci sont représentées à la figure suivante. Les relations d'héritage, de composition et d'agrégation y sont illustrés par un UML. Il est à noter que certaines méthodes (getters et setters) ont été ignorées dans ce diagramme pour alléger sa lecture. Par exemple, des méthodes tel que setTitre(), setTexte(), setNote() et setAuteur() seraient utilisées pour modifier ces attributs privés à travers la méthode ajouterCommentaire() du patient.

Voici une description de l'association entre les classes:

La classe patient et infirmière sont héritées de la classe utilisateur puisqu'un patient et une infirmière sont des sortes d'utilisateurs.

La classe profil est associée par composition à la classe utilisateur. Ceci s'explique par le fait que la destruction d'un utilisateur entraînerait la destruction de son profil dans le système. De plus, la construction d'un utilisateur entraîne la construction de son profil.

L'application de paiement est externe au système. Celle-ci est appelée par la classe rendez-vous à travers la méthode rdvComplete() et elle est associée à la carte de crédit du patient et au compte chèque de l'infirmière.

La carte de crédit est en relation de composition avec le patient car si le patient est enlevé du système, sa carte devra elle aussi être enlevée du système. Par la même logique, la classe CompteCheque est un composant de la classe Infirmière.

Le rendez-vous a une relation de composition avec le patient car c'est lui qui crée cet objet avec la méthode ajouterRendezVous(). Le rendez-vous sera alors initialisé avec un statut en attente. De plus, si le patient décide de se désinscrire de ce système, son rendez-vous sera aussi enlevé du système.

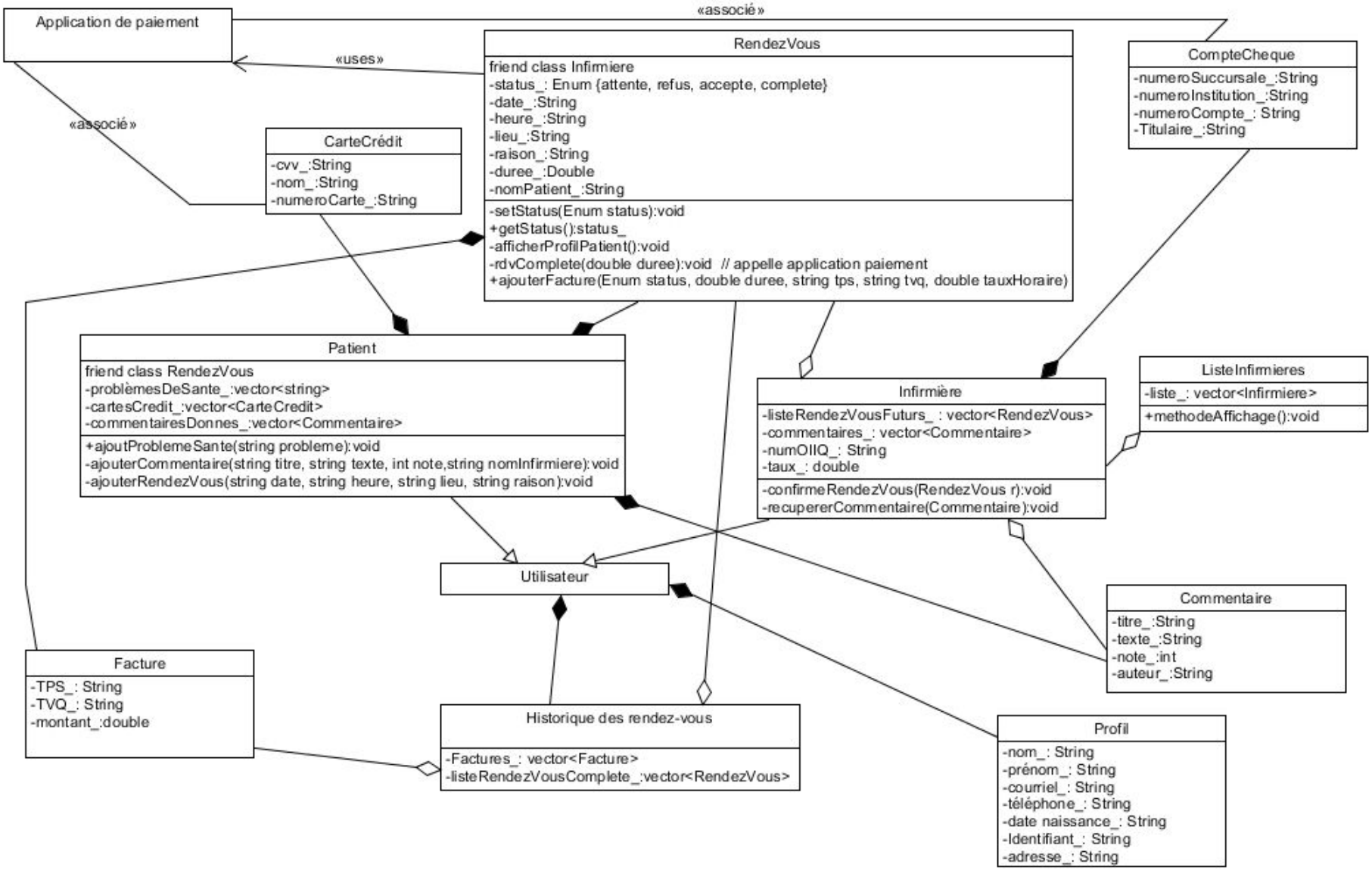
La facture est en relation de composition avec le rendez-vous car celui-ci est créé à l'étape finale du rendez-vous. La méthode rdvComplete() appelle alors la méthode ajouterFacture() qui crée un objet de type Facture en utilisant les informations entrées en paramètre. Si le rendez-vous est détruit, sa facture sera détruite.

Le commentaire est en relation de composition avec le patient. En effet, c'est le patient qui crée le commentaire à l'aide de la méthode ajouterCommentaire(). Si le patient se désinscrit du système, son commentaire sera alors détruit.

La classe historique de rendez-vous est en relation d'agrégation avec la classe facture puisqu'elle contient un vecteur de factures mais que sa destruction n'entraînera pas la destruction des factures. La même logique s'applique pour l'association d'agrégation entre la classe historique de rendez-vous et RendezVous.

L'infirmière est composée d'objets de la classe commentaire et RendezVous. Par contre, la destruction de l'infirmière n'entraîne pas la destruction de ces derniers. Il s'agit donc d'une relation d'agrégation. Par exemple, si une infirmière quittait, ses rendez-vous passés seraient toujours disponibles dans le système via l'historique des rendez-vous et ses rendez-vous futurs pourraient être repris par une autre infirmière.

Similairement, la classe ListeInfirmieres contient des infirmières. Une relation d'agrégation lie ces deux classes.


Application de paiement
«associé»
«uses»
«associé»
RendezVous
friend class Infirmiere
-status_: Enum {attente, refus, accepte, complete}
-date_:String
-heure_:String
-lieu_:String
-raison_:String
-duree_:Double
-nomPatient_:String
-setStatus(Enum status):void
+getStatus():status_
-afficherProfilPatient():void
-rdvComplete(double duree):void // appelle application paiement
+ajouterFacture(Enum status, double duree, string tps, string tvq, double tauxHoraire)
CompteCheque
-numeroSuccursale_:String
-numeroInstitution_:String
-numeroCompte_: String
-Titulaire_:String
CarteCrédit
-cvv_:String
-nom_:String
-numeroCarte_:String
Patient
friend class RendezVous
-problèmesDeSante_:vector<string>
-cartesCredit_:vector<CarteCredit>
-commentairesDonnes_:vector<Commentaire>
+ajoutProblemeSante(string probleme):void
-ajouterCommentaire(string titre, string texte, int note,string nomInfirmiere):void
-ajouterRendezVous(string date, string heure, string lieu, string raison):void
Infirmière
-listeRendezVousFuturs_ : vector<RendezVous>
-commentaires_: vector<Commentaire>
-numOIIQ_: String
-taux_: double
-confirmeRendezVous(RendezVous r):void
-recupererCommentaire(Commentaire):void
ListeInfirmieres
-liste_: vector<Infirmiere>
+methodeAffichage():void
Utilisateur
Facture
-TPS_: String
-TVQ_: String
-montant_:double
Historique des rendez-vous
-Factures_: vector<Facture>
-listeRendezVousComplete_:vector<RendezVous>
Commentaire
-titre_:String
-texte_:String
-note_:int
-auteur_:String
Profil
-nom_: String
-prénom_: String
-courriel_: String
-téléphone_: String
-date naissance_: String
-Identifiant_: String
-adresse_: String