BUREAU D'ETUDE – SIMULATION DE CURE THERMALE

UV - 5.8 - Ingénierie Systèmes

RÉSUMÉ

Ce document présente la simulation événementielle de la cure thermale de M Jabbalehut sur Tatooine. Elle présente une étude de son fonctionnement réalisée par les spécialistes de la société DarkForce.

Fabrice LALLEMENT - Noëlie RAMUZAT - Rémi RIGAL

Table des matières

Tableaux

Table des figures

OBJECTIF DE L'ETUDE

Notre étude a pour but de déterminer l'origine des insatisfactions des clients de la cure thermale de M Jabbalehut et de proposer des solutions pour améliorer leur contentement. Les critiques des curistes se portent sur le temps d'attente avant leurs ateliers ainsi que sur la mauvaise planification de leurs créneaux, les amenant parfois à ne pas pouvoir les réaliser. M Jabbalehut est prêt à diminuer la fréquentation maximale par jour au sein de sa cure mais préfèrerait d'autres alternatives.

Afin de réaliser l'étude demandée, notre société DarkForce décide de simuler de façon événementielle la cure thermale. Le logiciel créé se doit d'être flexible afin de pouvoir couvrir différents types de scénario et tester plusieurs configurations possibles pour permettre de choisir les solutions optimales aux problèmes des curistes. De plus notre outil se doit d'être adaptable à d'autres contextes afin de pouvoir le réutiliser pour d'autres types d'institut en difficulté. L'implémentation du logiciel de simulation se fait en langage Java.

Notre étude se penche en particulier sur les temps d'attente des clients au sein de l'institut ainsi qu'à la réussite de leurs ateliers. Ce dernier point est modélisé par un nombre de point que le client est susceptible d'améliorer en effectuant jusqu'au bout ses traitements.

ANALYSE DU PROBLEME

Exigences

ID	Intitulé	Type
E001	Simulation modulaire, configurable et adaptable.	Fonction
E002	L'institut accepte jusqu'à 180 curistes par jour.	Contrainte
E003	Il est ouvert du Lundi au Samedi inclus de 7h à 14h durant toute la période d'activité de Mars à Septembre.	Contrainte
E004	L'affluence à l'institut est variable selon les mois.	Fonction
E005	L'institut propose aux curistes différents ateliers.	Fonction
E006	Une cure dure 3 semaines consécutives pour chaque personne et doit être réalisée durant 3 années consécutives.	Contrainte
E007	Une cure commence toujours le Lundi matin.	Contrainte
E008	Les curistes ont entre 3 et 6 ateliers différents quotidiens à réaliser	Contrainte
E009	L'atelier Filiforme ouvert de 10h à 13h, tous les ateliers ouvrent de 7h15 à 14h00.	Contrainte
E010	Chaque zone a un responsable	Fonction

E011	Un atelier a une efficacité	Fonction
E012	Certains ateliers sont à heures fixées au curiste au préalable, d'autres à horaires libres.	Contrainte
E013	Deux types de files d'attentes existent pour les ateliers	Contrainte
E014	Un curiste qui a raté son créneau peut demander au responsable de zone s'il y aurait un créneau de disponible	Fonction
E015	Un responsable peut se trouver de manière équiprobable dans une autre zone	Contrainte
E016	Les ateliers peuvent tomber en panne.	Contrainte
E017	Présence de curistes peu scrupuleux	Contrainte
E018	Un curiste ne peut pas savoir les ateliers ayant des places libres.	Contrainte

TABLEAU 1 – EXIGENCES

Moteur de simulation

Dans le cadre de ce bureau d'étude, nous avons fait le choix de reconstruire un moteur de simulation intégralement. Cette volonté a plusieurs raisons. Tout d'abord cela nous permettait d'avoir un contrôle intégral sur le fonctionnement interne du moteur. De plus, d'un point de vue pédagogique, cela nous a permis de comprendre de manière plus approfondie les choix intrinsèques à un moteur de simulation (e.g. différence entre temps réel et événementiel) tout en nous permettant d'améliorer nos connaissances en Java. La structure de notre moteur de simulation se base sur trois éléments fondamentaux :

- un agenda contenant l'ensemble des événements prévus et se chargeant de les trier dans l'ordre chronologique,
- un scénario visant à fournir les particularités du problème considéré, il doit donc faire le lien entre moteur et données de simulation,
- le moteur en lui-même qui à partir de l'agenda et du scénario fait avancer la simulation, c'est lui qui a pour tâche de fournir le même temps logique à tous les éléments de la simulation ainsi que d'implémenter le générateur de nombre aléatoire ayant une seed contrôlable.

IMPLEMENTATION DU PROBLEME CONSIDERE

Notion d'entité

Les entités de simulation sont des éléments ayant un rôle prépondérant dans la simulation. L'une de leur caractéristique majeure est leur persistance dans le temps, par conséquent il est important de pouvoir les gérer finement. Elles suivent donc un cycle de vie strict permettant de gérer de manière séquentiel leur interaction avec le monde. Le cycle de vie est décrit dans les tableaux et le schéma suivant :

Etat	**Description**
NONE	L'entité en cours de constitution (constructeur).
BORN	L'entité est créée mais ne peut poster d'événements.
IDLE	L'entité est initialisée et peut poster des événements
ACTIVE	L'entité est en cours de traitement des événements qui la concerne.
DEAD	L'entité n'a plus de raisons d'exister. Elle n'est plus connue du moteur de simulation.

Transition	**Description**
Constructor	Invocation du constructeur par le programme principal. Les paramètres du constructeur représentent les données techniques structurantes de l'entité.
Initialize	Invocation par le moteur. Initialisation de l'entité. Les paramètres de l'initialisation représentent des données ne structurant pas l'entité.
Activate	Invocation par le moteur. A l'activation, les événements de l'entité peuvent être postés à partir de cet événement.
Pause	Invocation par le moteur. La Pause peut être invoquée suite à demande d'utilisateur (optionnel) via le moniteur.
Deactivate	Invocation par le moteur. Désactive en cascade les autres entités dont elle est parente. Retire les événements ultérieurs à l'événement présent de l'échéancier.
Terminate	Invocation par le moteur. L'entité enlève toute référence à d'autres. Ceci permet au garbage collector de vider la mémoire. L'entité ne doit plus être référencée par aucun objet.

TABLEAU 2 – ETATS ET TRANSITIONS DU MOTEUR DE SIMULATION

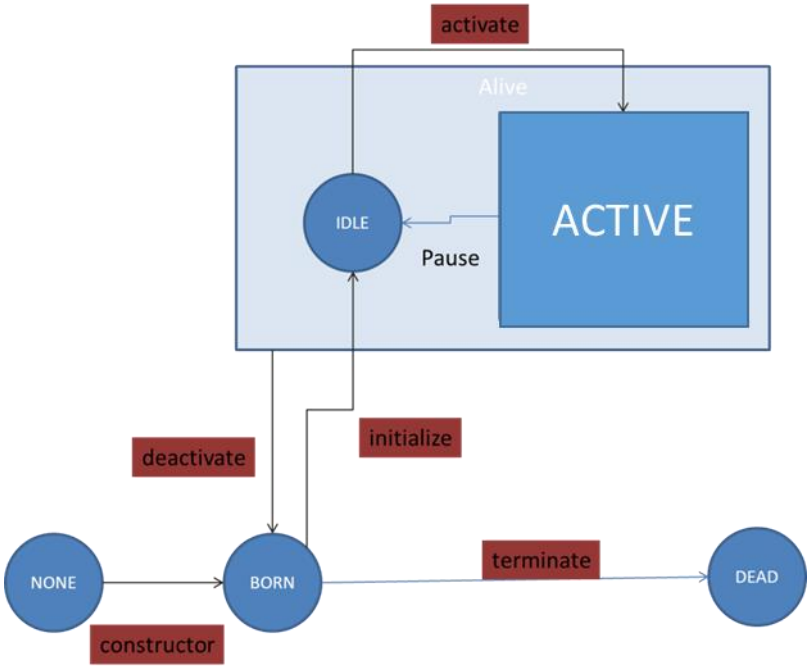

FIGURE 1 – MACHINE A ETATS DU MOTEUR

Dans notre implémentation actuelle, l'intégralité des options rendues disponibles par les entités ne sont pas encore tirées à profit. Par exemple, un utilisateur ne peut pas interagir avec la simulation pour mettre en pause certaines entités. Dans notre vision du problème, les entités sont les classes suivantes :

- l'établissement,
- les ateliers,
- les patients,
- les cures,
- les managers

Etablissement

Entité	SpaResort
Type d'Entité	Permanente
Variables d'états	List<Patient> patients Manager[] managers
Variables statistiques/de scrutation	Aucune
Paramètres techniques et Données d'initialisation	List<Month> openingMonths List<DayOfWeek> openingDays LocalTime[][] openingHours Treatment[] treatments int maxClients float[] inflowMonth
Evènements	OpenSpaEvent CloseSpaEvent
Comportement	Aucun
Processus Stochastique	Aucun

TABLEAU 3 – ENTITE ETABLISSEMENT

Explication

La cure thermale est représentée sous forme de classe sous l'appellation SpaResort. Cette classe a pour vocation de regrouper les caractéristiques fondamentales de la cure à la fois en termes d'infrastructure (e.g. ateliers disponibles) que de commerce (e.g. fréquentation mensuelle).

Etat

List<Patient> patients	Liste chaînée non ordonnée recensant l'intégralité des patients ayant fréquentés le spa
Manager[] managers	Tableau recensant tous les managers de l'établissement qui s'occupent d'un atelier chacun

Initialisation

La plupart des données caractéristiques de la cure thermale étant vouée à être constantes, elles sont définies en tant que « final » lors de la construction de l'instance. L'un des rôles prépondérants de cette entité lors de la simulation, outre fournir des données chiffrées comme la distance entre ateliers, consiste à initialiser au début les jours d'ouvertures du spa dans l'agenda du moteur de simulation. Cette initialisation respecte les hypothèses simplificatrices actuellement considérées.

Les données d'initialisations de l'institut sont les suivantes :

- Taux d'affluence des patients (E004)

Période	Mars	Avril	Mai	Juin	Juillet	Aout	Septembre
Affluence en % de la capacité maximale	50	60	70	80	95	90	65

TABLEAU 4 – TAUX D'AFFLUENCE DES PATIENTS

- Distance entre les ateliers

Durée (min)	Zone des jets filiformes	Zone des douches	Zone Bains à jets ancien	Zone des soins du visage	Zone des étuves	Zone Bains à jets modernes	Zone des terres chaudes	Zone non utilisée
Zone des jets filiformes	0	1	2	4	1	2	3	7
Zone des douches		0	1	2	2	2	4	6
Zone Bains à jets ancien			0	1	3	3	3	5
Zone des soins du visage				0	4	4	2	6
Zone des étuves					0	1	2	8
Zone Bains à jets modernes						0	1	8
Zone des terres chaudes							0	8
Zone non utilisée								0

TABLEAU 5 – DISTANCE ENTRE LES ATELIERS

Ateliers

Entité	Treatment
Type d'Entité	Permanente
Variables d'états	boolean withAppointment boolean isOrganizedWaiting boolean broken List<Patient> waitingQueue List<Patient> currentPatients List<LocalTime> appointmentTimes
Variables statistiques/de scrutation	int durationWaitedPerDay int patientsCuredPerDay
Paramètres techniques et Données d'initialisation	int id String name TreatmentType type Manager manager String openHour String closeHour boolean withAppointment int maxPatientsWorking int duration int maxPoints boolean isOrganizedWaiting int maxPatientsWaiting int failureMeanPerDay int failureSTDD int repairMeanDuration
Evènements	FailureEvent RepairEvent AvailabreTreatmentEvent
Comportement	addCurrentPatients() addWaitingQueuePatient() removeCurrentPatients() removeWaitingQueuePatient()
Processus Stochastique	Loi exponentielle pour défaillance des ateliers

TABLEAU 6 – ENTITE ATELIER

Explication

Les ateliers disponibles pour les patients sont modélisés par la classe Treatment. Les ateliers étant définis à l'avance et ne souffrant pas de changements, nous avons convenu de réaliser une énumération. L'avantage est de disposer d'une base immuable pour les ateliers tout en ayant la possibilité d'en rajouter simplement (E005).

Les ateliers ont à gérer leur file d'attente ainsi que les patients actuellement en cours de traitement. Chaque atelier possède un manager qui se trouve possiblement dans la zone de ce dernier. La dernière partie importante réside dans la gestion des éventuels problèmes entraînant la fermeture temporaire des ateliers (E016).

Initialisation

Lors de l'initialisation, chaque atelier génère en fonction de sa durée des horaires de rendez-vous possible et planifie le premier aléa qui entraînera une fermeture temporaire de l'atelier. (E009)

Les données d'initialisations des ateliers sont les suivantes (E003, E009, E011, E012, E013) :

Zones	Zone des terres chaudes	Zone des jets filiformes	Zone des étuves	Zone Bains à jets modernes	Zone Bains à jets ancien	Zone des douches	Zone des soins du visage
Fréquentation	planifié	libre	planifié	libre	libre	libre	libre
Plage d'ouverture	7h15-14h	10h-13h	7h15-14h	7h15-14h	7h15-14h	7h15-14h	7h15-14h
Nb de places	6	4	6	8	9	8	8
Durée (min)	20	5	15	20	20	10	10
Efficacité	20	30	15	15	10	10	5
Type file d'attente	organisé	non organisé	organisé	organisé	organisé	non organisé	Non organisé
Taille file	10	10	6	10	15	8	5

TABLEAU 7 – DONNEES D'INITIALISATION DES ATELIERS

Etat

boolean withAppointment	Indique si atelier à horaire fixe / libre
boolean isOrganizedWaiting	Indique si la file d'attente est organisée
boolean broken	Indique si l'atelier est en cours de réparation
List<Patient> waitingQueue	Liste les patients actuellement en train d'attendre
List<Patient> currentPatients	Liste les patients actuellement en soin
List<LocalTime> appointmentTimes	Liste les horaires possibles de rendez-vous

TABLEAU 8 – VARIABLES D'ETAT DES ATELIERS

Patients

Entité	Patient
Type d'Entité	Temporaire
Variables d'états	isFair
Variables statistiques/de scrutation	waitedDuration
Paramètres techniques et Données d'initialisation	int id boolean isFair int startYear int startWeek
Evènements	CreatePatientsEvent PatientArrivalEvent SearchForActionEvent ArrivedTreatmentEvent EndTreatmentEvent AppointmentTimeoutEvent LeaveSpaEvent PlanNewAppointmentEvent
Comportement	Aucun
Processus Stochastique	Aucun

TABLEAU 9 – ENTITE PATIENT

Explication

La classe Patient représente les curistes. Ils ont une durée de vie limitée dans le temps car ils sont supprimés en mémoire une fois qu'ils ont fini leur cure.

Initialisation

Lors de l'initialisation, les patients se voient attribuer une cure aléatoirement suivant les répartitions définies dans le cahier des charges. Il peut être honnête ou peu scrupuleux selon une probabilité donnée par le cahier des charges (E017). Un patient possède un compteur qui accumule des points en fonction du temps passé dans les ateliers. Ils peuvent replanifier un rendez-vous manqué avec le manager de l'atelier (événement *PlanNewAppointmentEvent*, E014, E018) mais pas seul car ils ne connaissent pas les disponibilités des ateliers.

Etat

boolean isFair	Indique si le patient est "peu scrupuleux"

TABLEAU 10 – VARIABLES D'ETAT DES PATIENTS

Cure

Entité	Cure
Type d'Entité	Temporaire (Entité enfant de Patient)
Variables d'états	List<Float> doneTreatments
Variables statistiques/de scrutation	int currentPoints int currentPointsThisDay int currentPointsThisYear
Paramètres techniques et Données d'initialisation	Patient owner int startYear int startWeek
Evènements	Aucun
Comportement	Géré par Patient
Processus Stochastique	Attribution du nombre d'ateliers par probabilité prédéfinie

TABLEAU 11 – ENTITE CURE

Explication

Une instance de Cure est attribuée à chaque patient lors de sa création. Cette cure définit le nombre d'ateliers journaliers (entre 3 et 6 : E008) et les horaires des rendez-vous sur 3 semaines pendant 3 ans (E006, E012). Elle définit la semaine de commencement de la cure, un lundi matin et s'assure que chaque année le début de la cure soit un lundi et sur une semaine pleine (E007).

Les horaires de rendez-vous sont calculés pour ne pas en proposer plus qu'un atelier ne peut accepter de patients pendant une période donnée. Qui plus est, ils sont suffisamment espacés dans le temps pour qu'il n'y ait pas de recouvrement entre plusieurs rendez-vous pour un même patient.

Etat

List<Float> doneTreatments	Indique pour chaque atelier à réaliser, le pourcentage de complétion (entre 0 et 1).

TABLEAU 12 – VARIABLES D'ETAT DE CURE

Managers

Entité	Managers
Type d'Entité	Permanente
Variables d'états	bool isInTreatmentZone Treatment locationInSpa
Variables statistiques/de scrutation	Duration durationInZone LocalDateTime nextPatientHour
Paramètres techniques et Données d'initialisation	Treatment treatment
Evènements	MoveZoneEvent
Comportement	addPatientAppointment()
Processus Stochastique	Loi uniforme pour le déplacement dans une zone

TABLEAU 13 – ENTITE MANAGER

Explication

La classe Manager représente les responsables des ateliers du spa. Ils restent inchangés durant toute la simulation et possèdent chacun un atelier à gérer.

Initialisation

Les managers sont instanciés avec leur atelier à gérer (E010). Ils connaissent les heures rendez-vous libres de leur atelier (s'il est à rendez-vous) par le biais de *List<LocalTime> appointmentTimes* de la classe Treatment. Ils peuvent donc replanifier un rendez-vous à un patient (*addPatientAppointment)* en créant un événement *AppointmentTimeoutEvent* au patient et en enlevant cet horaire à la liste des rendez-vous libres (E014, E018). Ils sont présents au sein de leur atelier (zone) à chaque fois qu'un patient commence leur atelier pour au moins deux minutes (*durationInZone*). Puis ils peuvent se déplacer dans tout le spa suivant une loi uniforme déclenchant l'événement *MoveZoneEvent* (E015).

Etat

bool isInTreatmentZone	Indique si le manager se trouve dans sa zone. Si oui un patient peut essayer de replanifier un rendez-vous, sinon non.
Treatment locationInSpa	Emplacement du manager dans le spa

TABLEAU 14 – VARIABLES D'ETAT DES MANAGERS

MODELISATION DU SYSTEME

Hypothèses simplificatrices

Certaines hypothèses simplificatrices ont été utilisées pour la modélisation de ce bureau d'étude. Ces hypothèses permettent de simplifier grandement l'implémentation du système tout en n'influant selon nous que peu sur les résultats obtenus. En conséquence, aucune des hypothèses énumérées ici ne nous semble dénaturer l'analyse à posteriori des résultats.

Hypothèse	Cause
Le spa n'ouvre que par semaines entières	L'ouverture de la cure thermale lors de semaines partiellement non travaillées n'a pas de sens étant donné que les cures se comptent en semaine. Cela engendrerait de l'impossibilité de finir les cures sans que cela soit dû au fonctionnement interne de l'établissement.
Chaque année comporte le même nombre de semaines d'ouverture	Cette hypothèse ne diminue que très peu le temps d'ouverture de l'établissement. Elle permet par contre de simplifier le retour de patients d'une année sur l'autre. En effet, si une année comporte plus de semaines ouvrables que l'année suivante, il ne sera pas possible de refaire venir tout le monde sans dépasser l'affluence maximale.
Les managers ne sont pas actuellement implémentés	La présence de managers par atelier, bien que rajoutant du réalisme, apporte peu à la simulation si ce n'est des phases de recherche à l'aveugle de la part des curistes. La replanification des rendez-vous est donc simplifiée et cela reviendrait à supposer que le manager est perpétuellement présent.
Au bout d'un certain temps sans trouver de place libre, le client quitte le spa.	Pour éviter que des clients tournent continuellement dans le spa alors qu'il n'y a plus de place libre, on considère que les patients partent du spa au bout d'un certain temps sans trouver de place. Cela impacte leur contentement.
Aucune distinction n'est faite pour la fermeture des ateliers entre problème d'infrastructure ou maladie du personnel	Du point de vue du rendement de la cure thermale, peu importe la raison de fermeture d'un atelier. Nous avons donc choisi de nous focaliser sur la durée de fermeture. Qui plus est, bien qu'une loi modélisant une usure soit facile à trouver, il est bien plus compliqué de modéliser le fait de tomber malade.

TABLEAU 15 - HYPOTHESES SIMPLIFICATRICES

Modélisation du moteur de simulation

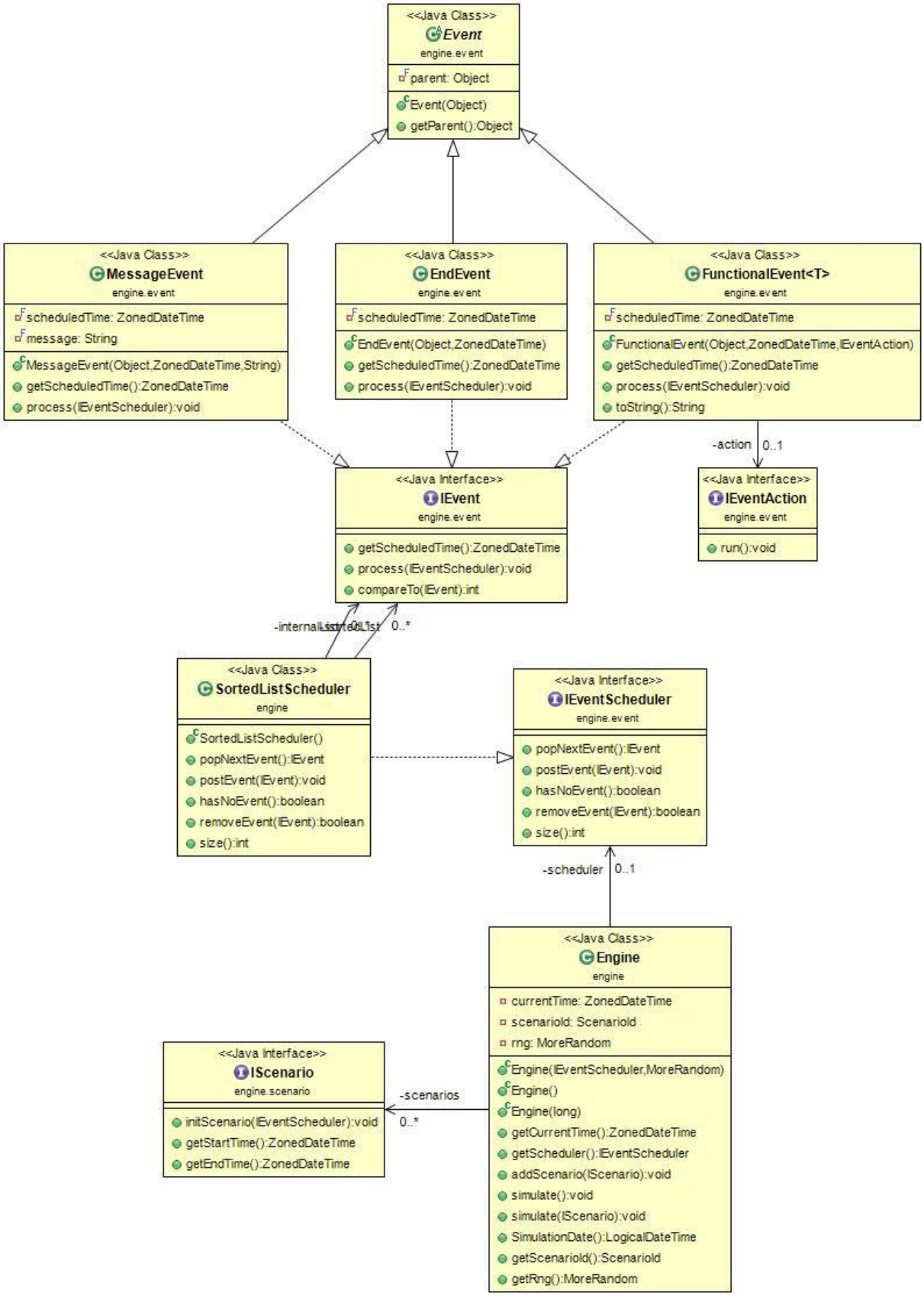

FIGURE 2 - DIAGRAMME DE CLASSE DU MOTEUR DE SIMULATION

Modélisation des classes

Établissement

Stochastique

L'établissement ne dispose d'aucune méthode implémentant des processus aléatoires.

Evénements

L'établissement possède deux événements *OpenSpaEvent* et *CloseSpaEvent* modélisant l'ouverture et la fermeture de l'institut.

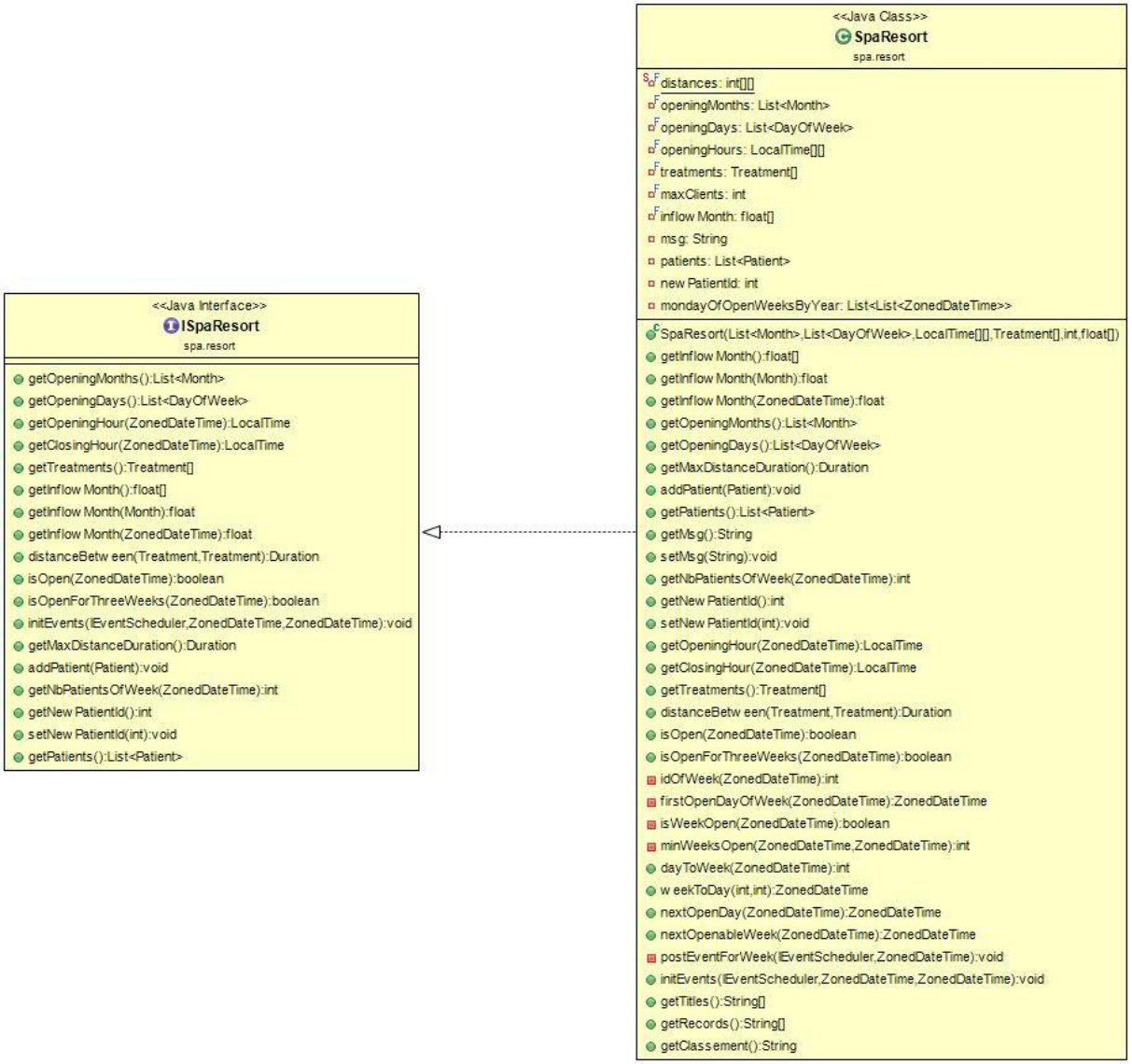

FIGURE 3 – DIAGRAMME DE CLASSE DE L'ETABLISSEMENT

Ateliers

Stochastique

Nous avons réfléchi à la manière de modéliser les aléas des ateliers. L'origine de ces fermetures étant principalement dû à de l'usure de matériel, nous voulions utiliser une loi représentant correctement l'usure. De ce fait, le risque se devait d'être faible au début avant d'arriver vers une valeur moyenne de défaillance à partir de laquelle la probabilité augmentait grandement. Une répartition uniforme n'était donc, par exemple, pas envisageable.

La première loi que nous avons voulu considérer était la loi de Weibull, couramment utilisée dans la modélisation des problèmes d'usure. Cependant, l'implémentation informatique était compliquée pour un gain négligeable en cohérence en comparaison d'autres lois. De ce fait, nous avons finalement opté pour une loi exponentielle facilement implémentable et fournissant des résultats similaires à la loi de Weibull.

En ce qui concerne la modélisation de la remise en marche des ateliers nous avons utilisé une loi uniforme.

Données d'initialisation à respecter lors du calcul des lois exponentielle et uniforme (E016) :

Données d'initialisation à respecter lors du calcul des lois exponentielle et uniforme (E016) :

Zones	Zone des terres chaudes	Zone des jets filiformes	Zone des étuves	Zone Bains à jets modernes	Zone Bains à jets ancien	Zone des douches	Zone des soins du visage
Nombre moyen de jour de panne d'un atelier par an	61	28	21	70	35	49	365
Temps de remise en marche moyen en jours	3	2	3	4	2	2	1

TABLEAU 16 - DONNEES CARACTERISTIQUES DES ATELIERS

Evènements

Les ateliers possèdent trois événements :

- *FailureEvent* modélisant la panne d'un atelier
- *RepairEvent* modélisant la réparation d'un atelier
- *AvailableTreatmentEvent* notifiant qu'une place s'est libérée dans un atelier et choisissant le prochain patient qui effectue l'atelier

Données d'initialisation à respecter lors de la libération d'une place dans un atelier :

- Comportement à respecter pour les curistes peu scrupuleux (E013)

Type file attente	Organisée	Non organisée
Temps en secondes laissé au curiste avant de prendre sa place	20	10
Fréquence de prise de place	1 fois sur 10	1 fois sur 4

TABLEAU 17 - REPARTITION DU TYPE DE FILE D'ATTENTE

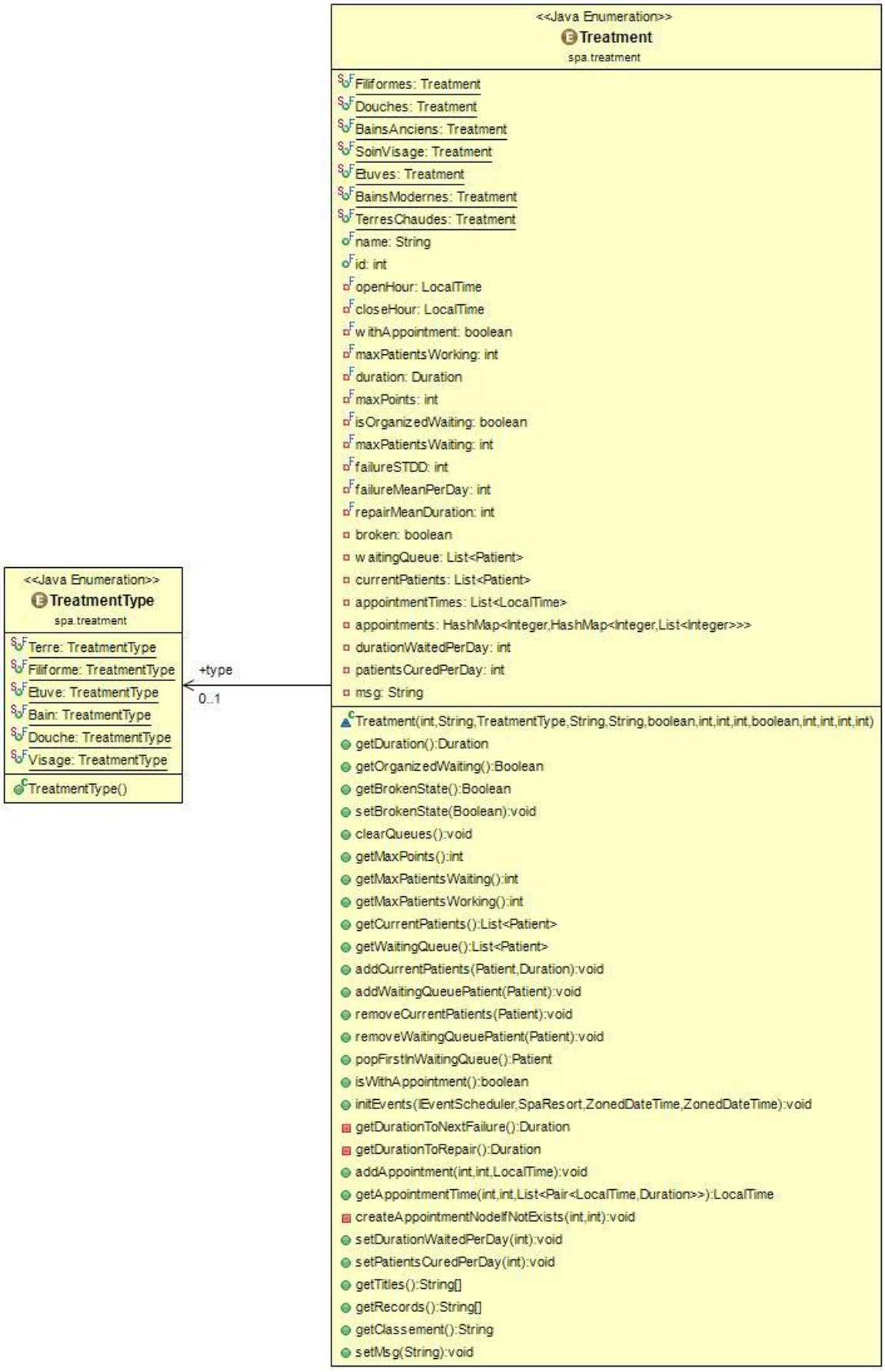

FIGURE 4 - DIAGRAMME DE CLASSE TREATMENT

Patients

Stochastique

Deux répartitions ne sont pas déterministes pour le patient, le choix des ateliers de sa cure ainsi que son honnêteté. Cependant, ces deux répartitions sont parfaitement connues d'un point de vue probabiliste et n'ont pas donc nécessité de réflexion sur la loi à implémenter pour les modéliser. Un simple tirage suffit.

Comportement

Un patient est créé le dimanche précédent le premier lundi de la première semaine de sa cure. Il possède une liste d'atelier à effectuer et connait ses rendez-vous.

- Il arrive le matin à 7h15 à l'accueil et commence à chercher un atelier sans rendez-vous à effectuer parmi sa liste.
    - S'il a cherché un atelier libre cinq fois sans en avoir trouvé il va se reposer dans la salle d'attente pendant une heure
    - Sinon, s'il trouve un atelier il s'y déplace
    - Sinon, il quitte le spa (journée de cure terminée)
- Le patient arrive dans un atelier, trois cas se présentent :
    - Il y a de la place, le patient commence le traitement, il gagne des points
    - Il y a de la place dans la file d'attente, son temps d'attente augmente
    - Il n'y a pas de place, il recommence à chercher
- Le patient finit un atelier, ses points sont comptabilisés, 2 cas se présentent :
    - Il cherche un nouvel atelier de libre
    - Son atelier a pris fin car c'est la fermeture : il quitte le spa

À tout moment de la journée le patient peut avoir un rendez-vous et ses activités en cours sont arrêtés. Il rejoint alors l'atelier du rendez-vous et effectue son traitement puis reprend sa journée.

De même, lorsque le spa ferme le patient arrête ses activités et quitte le spa.

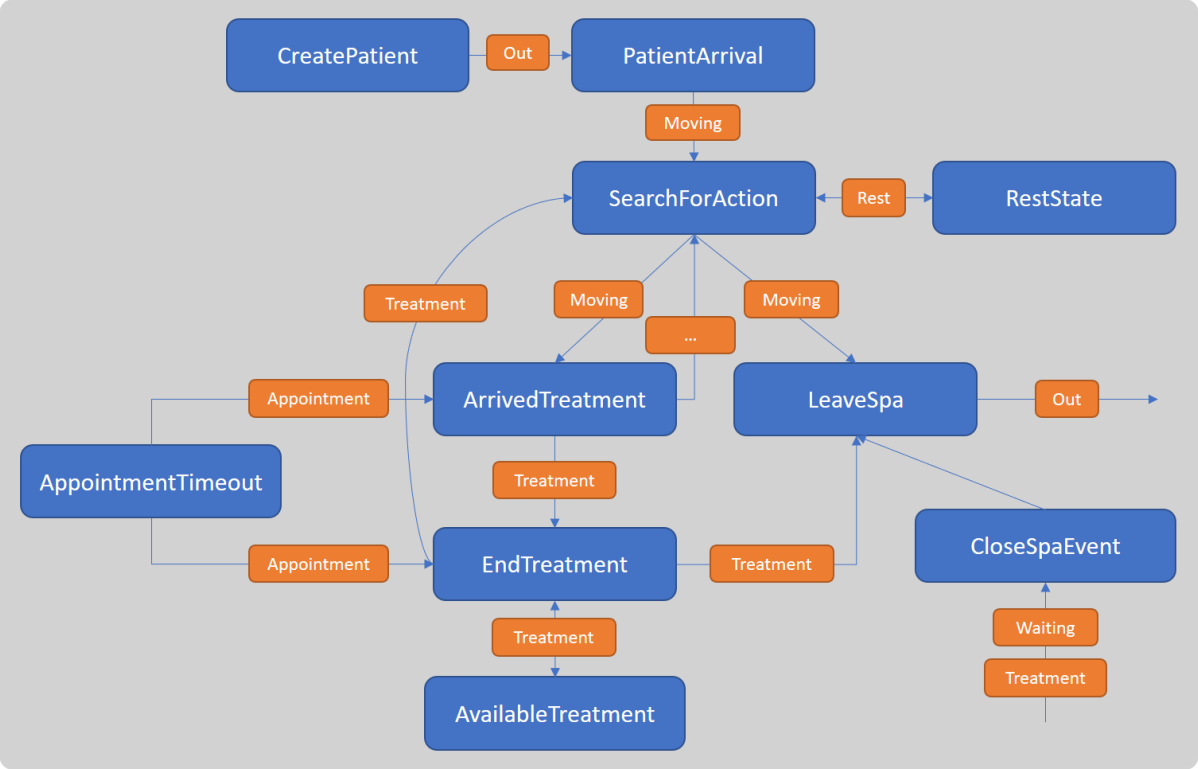

FIGURE 5 - MACHINE A ETATS DES CURISTES

Evènements

Les patients possèdent sept événements :

- *CreatePatient* modélisant la création d'un patient
- *PatientArrival* modélisant l'arrivée d'un patient dans le spa le matin
- *SearchForAction* modélisant la recherche de la prochaine action du patient : rejoindre un atelier, se reposer ou quitter le spa
- *ArrivedTreatmentEvent* modélisant l'arrivée d'un patient dans un atelier
- *EndTreatmentEvent* modélisant la fin d'un atelier par le patient
- *AppointmentTimeoutEvent* modélisant une alarme pour que le patient arrête son activité en cours pour rejoindre l'atelier de son rendez-vous
- *LeaveSpaEvent* modélisant le départ du patient du spa le soir

Données d'initialisation à respecter lors de la création d'un nouveau patient :

- Répartition en pourcentage des curistes peu scrupuleux (E017)

% de curiste peu scrupuleux	5 %
Fréquence d'oubli des créneaux en ateliers	1 fois sur 20
Ecart type en minutes des délais pris par le curiste en atelier libre	2

TABLEAU 18 – REPARTITION DU NOMBRE DE PATIENTS PEU SCRUPULEUX

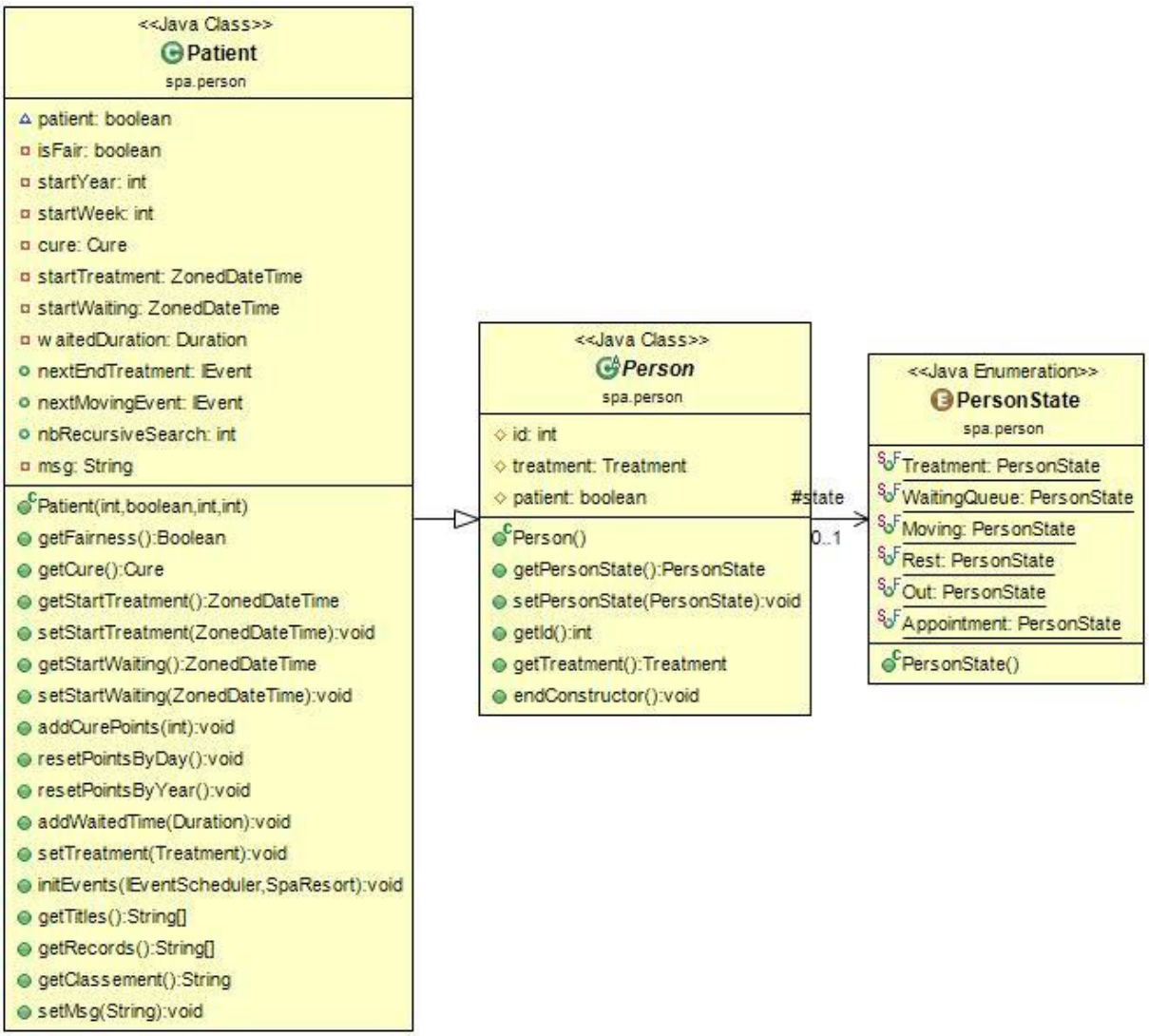

FIGURE 6 - DIAGRAMME DE CLASSE PATIENT

Cure

Stochastique

L'attribution du nombre de cures s'effectue par une loi de probabilité correspondant à une somme de lois uniformes. En effet, chaque possibilité de nombre d'ateliers dans la cure s'est vu affecter dans le cahier des charges un probabilité d'affectation rappelée ci-après.

- Répartition des curistes en fonction du nombre d'ateliers

Nombre Ateliers	3	4	5	6
Répartition en %	20	35	30	15

TABLEAU 19 - REPARTITION DU NOMBRE D'ATELIERS

Les événements

Afin de réaliser une simulation événementielle de la cure thermale nous implémentons une liste contenant les événements à venir. Cet agenda les trie par ordre chronologique en fonction du temps logique simulé. Tout au long de la simulation ces événements sont retirés de l'agenda puis exécutés. Ils sont identifiés et répartis entre les différentes entités comme suit :

Etablissement

OpenSpaEvent

Cet événement est créé pour chaque jour d'ouverture du spa à son heure d'ouverture (7h). Il permet de logger les horaires du spa et de vérifier le bon fonctionnement du moteur de simulation. Il notifie les managers du changement d'état du spa.

CloseSpaEvent

Cet événement est créé pour chaque jour de fermeture du spa lors de son heure de fermeture (14h). Il notifie les managers du changement d'état du spa. Un événement pour quitter le spa est ajouté pour chaque patient en cours de déplacement ou en train d'attendre.

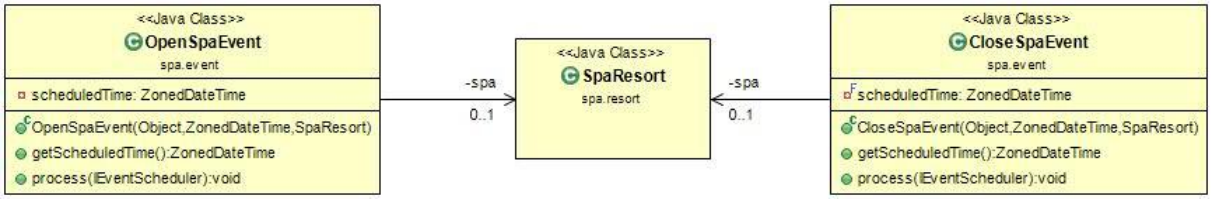

FIGURE 7 - DIAGRAMME DE CLASSE SPA AVEC EVENEMENTS

Ateliers

FailureEvent

Ces événements sont créés au début de la simulation, lors de la création des différents ateliers permettant le calcul de leurs jours de fermeture. Ils sont créés par atelier en fonction du jour et de l'heure renvoyés par la loi exponentielle simulant leur probabilité de panne. Cet événement met à jour la variable d'état *broken* de l'atelier pour signaler la panne de celui-ci et en informe le manager de l'atelier. Chaque patient effectuant le traitement ou attendant son tour se voit ajouter un événement de fin d'atelier au même horaire que le *failureEvent* afin d'évacuer les patients.

RepairEvent

Comme pour *failureEvent*, ces événements sont créés au début de la simulation, lors de la création des différents ateliers permettant le calcul de leur temps de réparation et après la création du *failureEvent* associé. Cet événement met à jour la variable d'état *broken* de l'atelier pour signaler la réparation de celui-ci et en informe le manager de l'atelier.

AvailableTreatmentEvent

Cet événement est créé lors du départ d'un patient d'un atelier sans rendez-vous après que ce patient ait fini son traitement. Il cherche le curiste suivant dans la file d'attente de l'atelier si elle n'est pas vide. Ce curiste est choisi en fonction de s'il est le premier dans la file d'attente et est assez rapide pour rentrer dans l'atelier ou s'il existe un patient malhonnête qui arrive à prendre sa place. Puis l'événement ajoute le patient choisi à la liste des patients en cours de traitement dans l'atelier et créer un événement fin d'atelier pour ce patient. L'horaire attribué à cet événement correspond à celui de la fermeture du spa si celle-ci intervient avant la fin prévue de l'atelier. Il instancie également l'horaire de commencement de l'atelier par le patient pour le calcul de ses points et calcule son temps d'attente dans la queue.

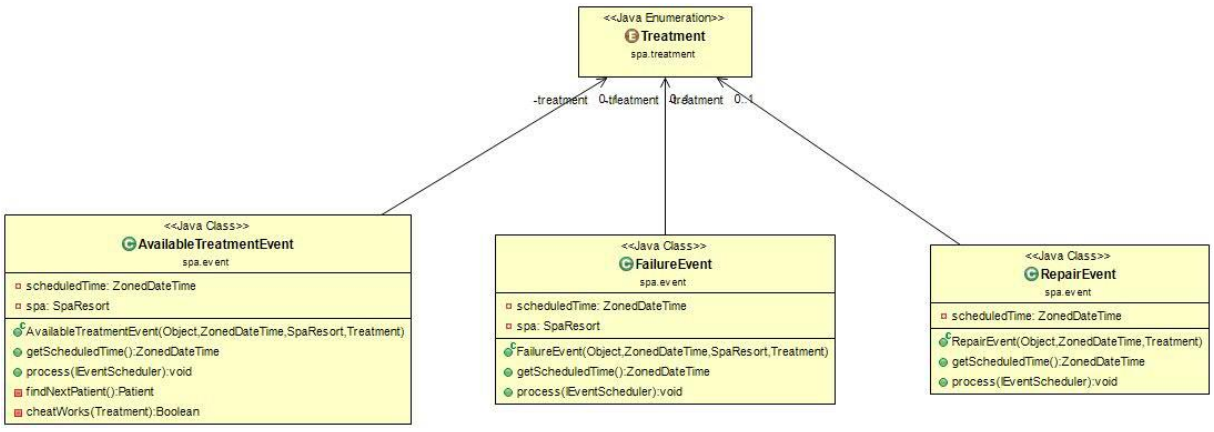

FIGURE 8 - DIAGRAMME DE CLASSE TREATMENT AVEC EVENEMENTS

Patient

CreatePatientsEvent

Cet événement est créé à chaque fin de semaine (dimanche) afin de créer les patients manquant pour atteindre le taux d'affluence correspondant au mois de l'année. Le patient créé par cet événement a 5% de chance d'être malhonnête et sa date de commencement de cure est instancié au lundi suivant. L'identifiant du patient est une variable de la classe qui s'incrémente à chaque création. L'initialisation des événements du nouveau patient est ensuite effectuée (prise de rendez-vous, arrivée dans le spa).

PatientArrivalEvent

Cet événement est créé lors de l'initialisation de chaque patient, pour chaque matin des neuf semaines sur les trois ans à venir. Il est déclenché à l'arrivée d'un client dans le spa à chaque jour de sa cure. Il remet à zéros la liste des ateliers de la journée déjà effectuées par le patient, change l'état du patient pour le mettre à "en mouvement" et créer un nouvel événement de recherche d'action à effectuer par le curiste.

SearchForActionEvent

Cet événement est créé suite à l'arrivée d'un patient dans le spa ou à la fin du traitement dans un atelier ou s'il n'y a plus de place dans un atelier et que le patient doit en chercher un autre. Cet événement change l'état du patient pour le mettre à "en mouvement". Il choisit le prochain atelier que le patient va faire ou s'il va en zone de repos ou s'il quitte le spa.

- Il élit un atelier en fonction de sa proximité, de s'il est avec rendez-vous et de s'il est en panne. Puis il créé l'événement arrivée dans un atelier pour le patient et l'atelier choisi.
- Il élit la zone de repos si le patient a cherché auparavant cinq fois un atelier libre (depuis le début de la journée ou de la dernière fois qu'il s'est reposé) sans en trouver un.
- Un événement pour quitter le spa est créé à la place s'il ne reste aucun atelier à effectuer au patient ou si l'heure de fin de traitement se produit après celui de la fermeture du spa (l'horaire de l'événement correspond alors à celui de la fermeture du spa).

ArrivedTreatmentEvent

Cet événement est créé suite à la recherche d'action par le patient et représente le patient arrivant dans un atelier après s'être déplacé.

- S'il y a de la place dans l'atelier il ajoute le patient dans la liste des curistes en cours de traitement, et créer un événement fin de traitement. L'horaire attribué à cet événement correspond à celui de la fermeture du spa si celle-ci intervient avant la fin prévue de l'atelier. Il change l'état du patient pour le mettre "en cours de traitement". Il instancie également l'horaire de commencement de l'atelier par le patient pour le calcul de ses points.
- S'il n'y a pas de place dans l'atelier mais de la place dans la file d'attente le patient est ajouté à cette dernière. Il change l'état du patient pour le mettre "en attente". L'événement instancie l'horaire de début d'attente pour calculer le temps d'attente des patients.
- Sinon il crée un nouvel événement de recherche d'action à effectuer par le curiste.

EndTreatmentEvent

Cet événement est créé suite à l'arrivée d'un patient dans un atelier ayant de la place ou suite à la sélection d'un patient dans la file d'attente. Il correspond à la fin du traitement du patient dans l'atelier donné. L'événement enlève le curiste de sa liste de patient en cours de traitement et calcule ses points en fonction du temps resté dans l'atelier. Il met à jour la liste des ateliers effectués par le patient ce jour-ci et créer un nouvel événement de recherche d'action à effectuer par le curiste sauf si ce dernier a un rendez-vous. Puis il crée un événement signalant que l'atelier à une place de libre.

AppointmentTimeoutEvent

Cet événement est créé lors de l'initialisation des événements des patients suite à leur création. Les rendez-vous des curistes pour les ateliers en proposant sont calculés au début de la première semaine pour les neuf semaines prévues dans la cure. Cet événement est prévu cinq minutes avant l'heure de rendez-vous et fonctionne comme une alarme qui signale au patient d'aller à son atelier. Il créé un événement de fin d'atelier si le curiste effectue un traitement ainsi qu'un événement d'arrivée dans l'atelier du rendez-vous pour le patient. L'événement change l'état du patient pour le mettre à "en rendez-vous".

LeaveSpaEvent

Cet événement est créé suite à la fermeture du spa qui amène les patients à quitter l'institut et à notifier les managers du changement d'état du spa. L'événement change l'état du patient pour le mettre à "est parti".

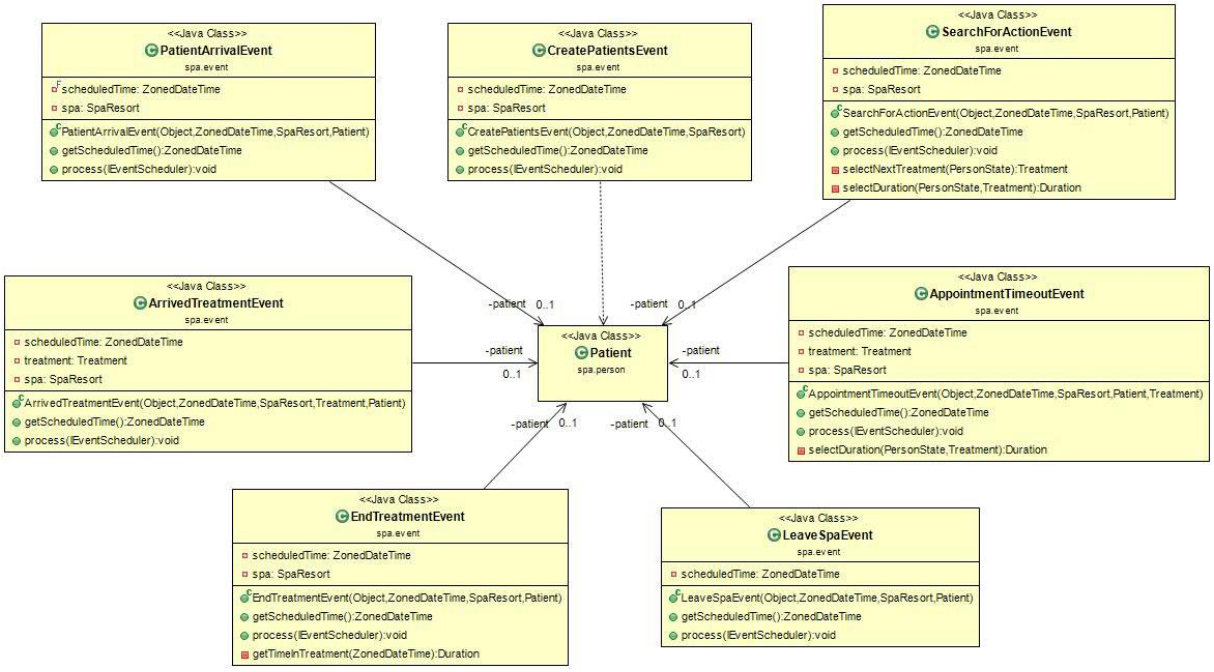

FIGURE 9 - DIAGRAMME DE CLASSE PATIENT AVEC EVENEMENTS

Scénario

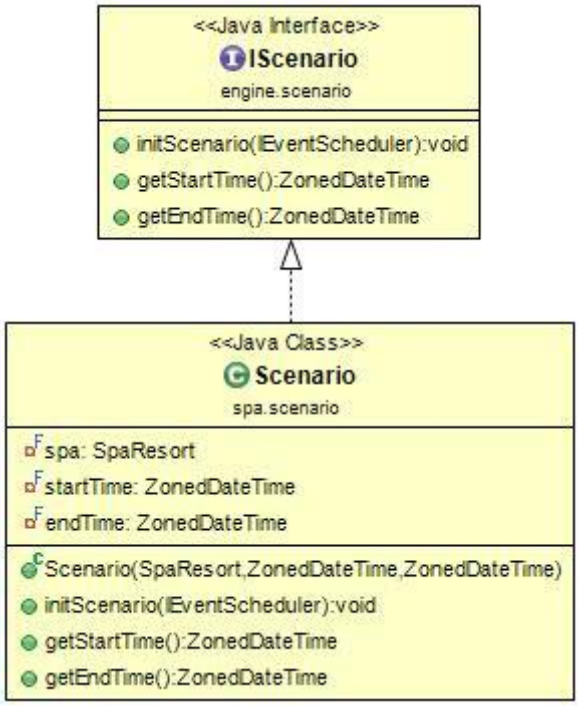

FIGURE 10 - DIAGRAMME DE CLASSE SCENARIO

Le scénario est défini par différentes informations inhérentes au fonctionnement de la simulation. Il a pour tâche de regrouper ces données pour le moteur de simulation. Ainsi, lors du lancement de notre simulation on initialise en premier lieu un scénario qui est ensuite ajouté à la liste de scénarios traitée par le moteur de simulation (E001).

```
Engine engine = new Engine();

List<Month> openingMonths = new ArrayList<>();
openingMonths.add(Month.APRIL);
openingMonths.add(Month.MAY);
openingMonths.add(Month.JUNE);
openingMonths.add(Month.JULY);
openingMonths.add(Month.AUGUST);

List<DayOfWeek> openingDays = new ArrayList<>();
openingDays.add(DayOfWeek.MONDAY);
openingDays.add(DayOfWeek.TUESDAY);
openingDays.add(DayOfWeek.WEDNESDAY);
openingDays.add(DayOfWeek.THURSDAY);
openingDays.add(DayOfWeek.FRIDAY);

LocalTime openTime = LocalTime.parse("07:00:00");
LocalTime closureTime = LocalTime.parse("14:00:00");
LocalTime[][] openingHours = {{openTime, openTime,  openTime, openTime, openTime, openTime,  openTime},
                              {closureTime, closureTime, closureTime, closureTime, closureTime, closureTime, closureTime}};

int maxPatients     = 180;
float[] inflowMonth = new float[] {0f, 0f, 0.5f, 0.6f, 0.7f, 0.8f, 0.95f, 0.9f, 0.65f, 0f, 0f, 0f};

Treatment[] treatments = {Treatment.BainsAnciens, Treatment.BainsModernes, Treatment.Douches, Treatment.Etuves,
        Treatment.Filiformes, Treatment.SoinVisage, Treatment.TerresChaudes};

SpaResort spa = new SpaResort(openingMonths, openingDays, openingHours, treatments, maxPatients, inflowMonth);

ZonedDateTime startTime = ZonedDateTime.parse("2018-01-01T00:00:00+01:00[Europe/Paris]");
ZonedDateTime endTime   = ZonedDateTime.parse("2019-01-01T00:00:00+01:00[Europe/Paris]");

Scenario scenario = new Scenario(spa, startTime, endTime);

engine.addScenario(scenario);
```

Dans notre cas, les contraintes du spa et de la simulation sont définies lors de la construction du scénario ce qui correspond à :

- les mois d'ouvertures (E003),
- les jours d'ouvertures (E003),
- les heures d'ouvertures (E003),
- les affluences mensuelles (E004),
- le nombre maximal de curistes (E002),
- les ateliers disponibles dans le spa (E005),
- les dates de début et de fin de simulation.

IMPLEMENTATION DU MODELE

La structure du projet est établie selon les dossiers suivants :

- **src** : code source du logiciel de simulation
- **conf** : paramètres du logger représentés en Json
- **doc** : documents décrivant le cahier des charges
- **log** : fichiers Excel des résultats de la simulation
- **Rapport** : rapport du projet de simulation de cure thermale
- **UML_Diagram** : Diagrammes de classe modélisés à partir du code java

Le code source du logiciel se décompose entre les parties suivantes :

- **engine** : Moteur de simulation
    - **event** : Événements du moteur
        - *EndEvent.java* : déclenche l'arrêt de la simulation
        - *Event.java* : classe abstraite des événements de la simulation
        - *FunctionalEvent.java* : étend *Event*, modèle des autres événements
        - *IEvent.java* : Interface java de *Event*
        - *IEventScheduler.java* : Interface java de l'agenda des événements
        - *MessageEvent.java* : Evènement permettant de logger des informations
    - **scenario** : Schéma du scénario à suivre
        - *IScenario.java* : Interface java définissant l'implémentation des scénarios
    - *Engine.java* : Classe définissant le fonctionnement du moteur : boucle de simulation, définition de la graine aléatoire, initialisation du scénario
    - *SortedListScheduler.java* : Classe définissant l'agenda des événements, trié
- **logger** : Wrapper du Logger défini dans le package simu_base_common
    - *IRecordableWrapper.java* : Wrapper de l'interface IRecordable, ajoute une méthode permettant de définir un message dans le logger
    - LoggerWrap.java : Permet de rajouter un message de description de l'événement loggé

- **spa**
    - **cure**
        - *Cure.java* : Classe définissant les cures des patients avec leurs points et leurs ateliers et les rendez-vous
    - **entity**
        - *IEntity.java* : Interface d'implémentation des entités
        - *Entity.java* : Classe abstraite implémentant les fonctions de changement d'état
    - **event** : Événements décrits dans la partie « Les événements »
    - **person**
        - *Patient.java* : Classe définissant les patients, héritant de *Person*,
        - *Person.java* : Implémentation du patient et gestion de la cure.
        - *PersonState.java* : Enumération des états possibles des patients
    - **resort**
        - *ISpaResort.java* : Interface d'implémentation de l'institut
        - *SpaResort.java* : Classe définissant le spa avec les distances entres les ateliers, les jours d'ouvertures et de fermetures, les horaires, les ateliers, le nombre maximal de client et l'affluence par mois
    - **scenario**
        - *Scenario.java* : implémentation de l'interface IScenario, contient les données temporelles de la simulation
    - **treatment**
        - *TreatmentType.java* : Définition des différents types de soin
        - *Treatment.java* : Enumération des ateliers à implémenter
    - *ScenarioTest.java* : Script de test des scénarios permettant de lancer la simulation
    - **UML** : Diagrammes de classe réalisés avec le plugin *ObjectAid*

PRESENTATION DES RESULTATS

Résultats et analyse de la simulation

Notre simulation a été effectuée sur deux ans, de 2018 à 2019 avec des mois d'ouvertures d'avril à septembre du lundi au samedi (nous avons oublié de simuler le mois de mars). Lors de cette simulation l'institut s'est occupé de 1131 patients. Nous nous intéressons au temps d'attente de ces derniers, à l'efficacité de leur cure (lié à leur nombre de point), à l'affluence dans la cure ainsi qu'au taux de panne des ateliers.

Population de l'institut

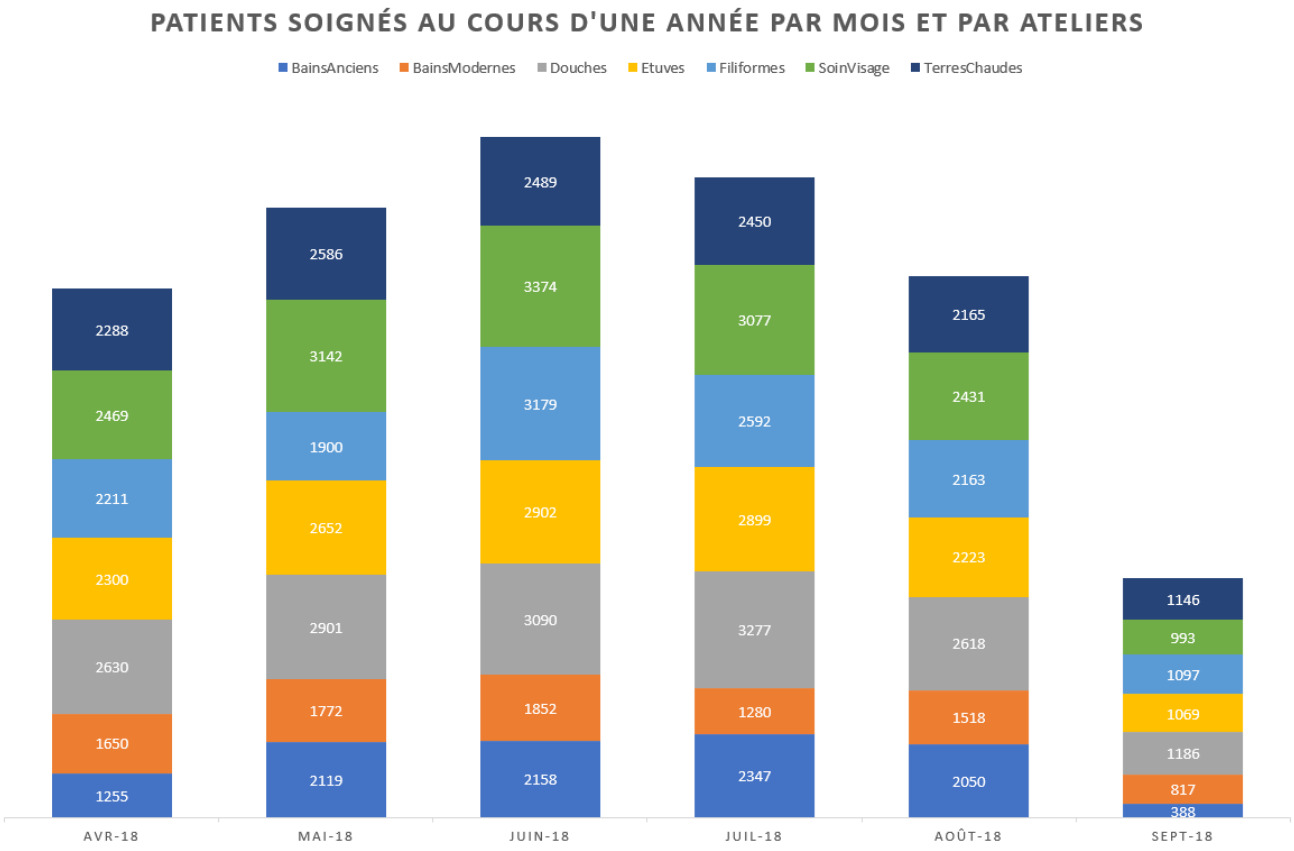

FIGURE 11 – EVOLUTION DU NOMBRE DE PATIENTS SOIGNES

Cet histogramme présente le nombre de patient soignés par mois par ateliers au cours d'une année. Nous remarquons bien l'évolution de l'affluence en fonction des mois de l'année, qui augmente en été pour beaucoup diminuer en septembre (qui a le plus faible taux d'affluence dans le cahier des charges). Nous observons que les douches et les soins du visages sont les ateliers les plus utilisés par les patients.

Indisponibilité des ateliers

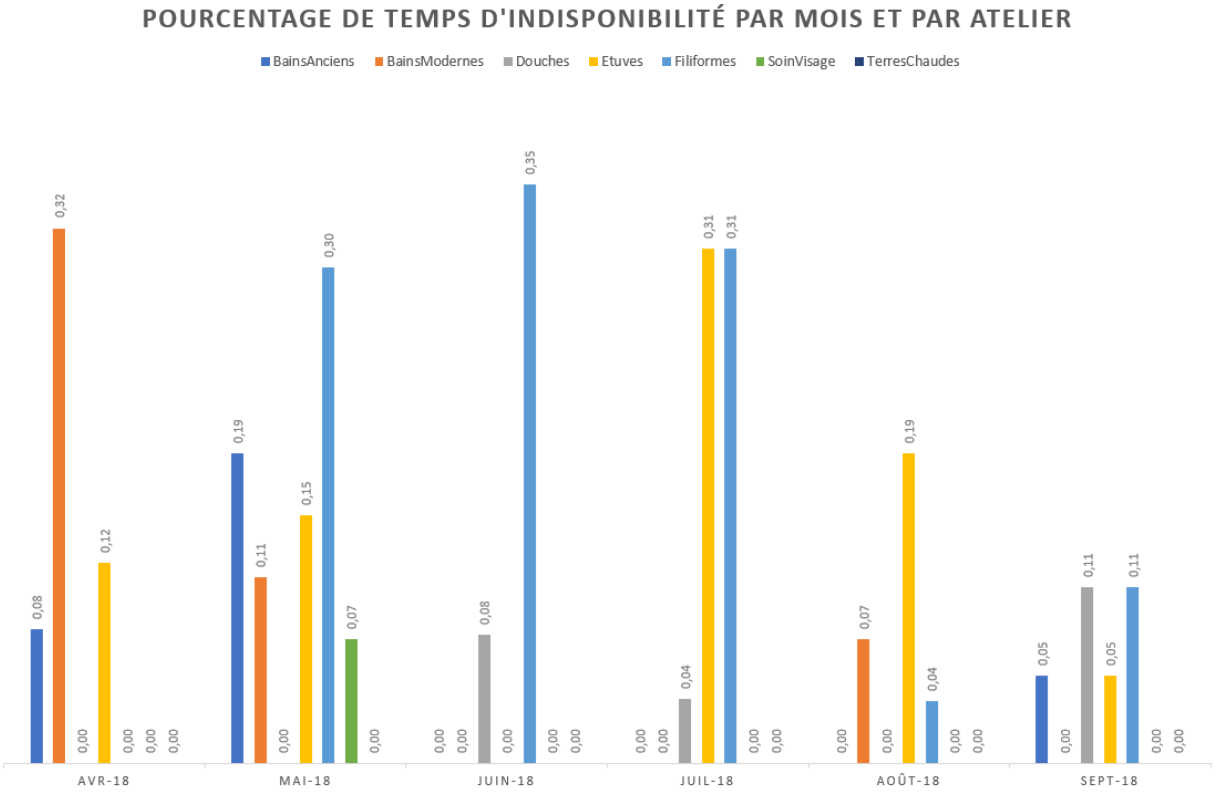

FIGURE 12 - INDISPONIBILITE MENSUELLE

Cet histogramme présente le temps d'indisponibilité des ateliers par mois durant une année. Nous remarquons que mis à part le mois d'avril les étuves et les jets filiformes sont les ateliers les plus sensibles, ils tombent le plus souvent en panne ce qui diminue l'efficacité de l'institut. Il serait intéressant pour M. Jabbalehut d'améliorer la maintenance de ses ateliers.

Attente des patients

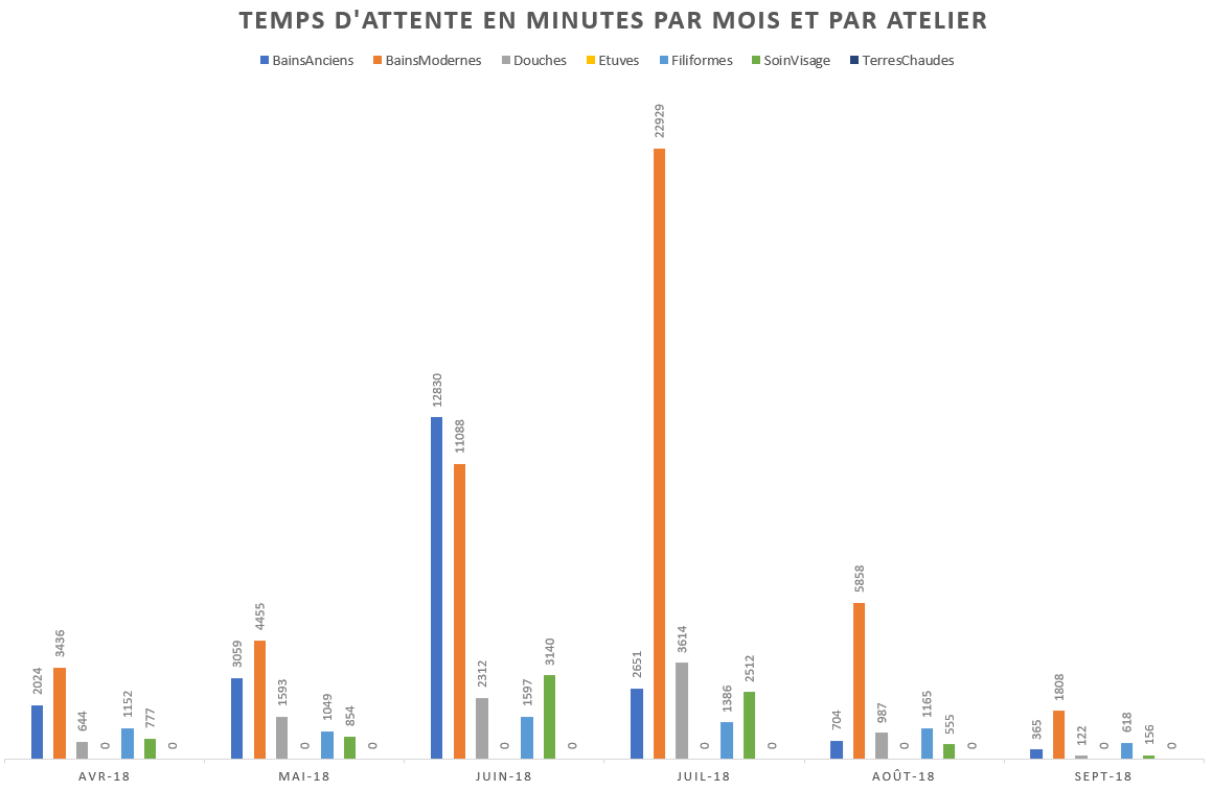

FIGURE 13 – TEMPS D'ATTENTE MOYEN INITIAL PAR ATELIER

Cet histogramme présente le temps d'attente des curistes en minutes par mois et par atelier durant la première année. Nous observons que les bains sont les ateliers critiques de l'institut avec un temps d'attente des patients de 15 jours au cumulé en juillet. Il est donc urgent de prendre une décision pour ce type d'atelier.

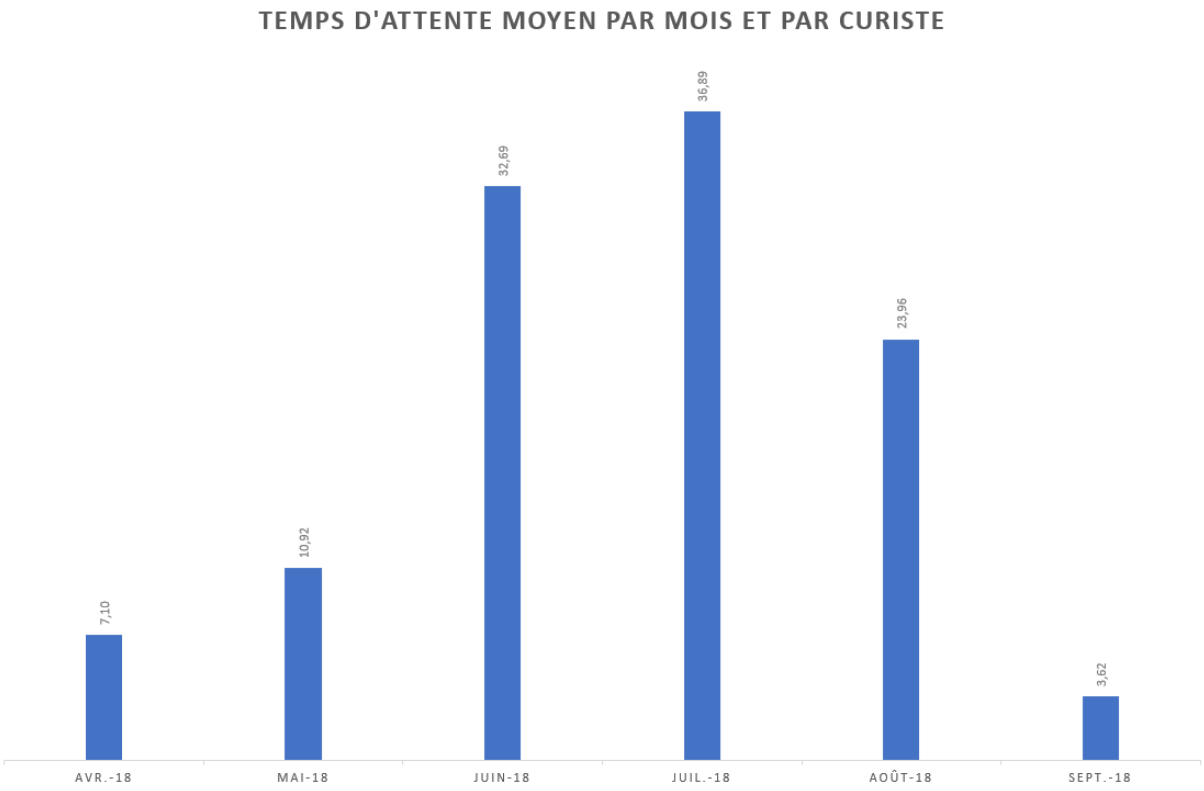

FIGURE 14 – TEMPS D'ATTENTE MOYEN INITIAL PAR CURISTE

Cet histogramme représente le temps d'attente moyen par mois et par curiste, nous retrouvons bien le mois de juillet comme pire mois de l'année avec un temps moyen de 37 minutes par curiste. En moyenne un patient attend 20 minutes par mois.

Efficacité de l'institut

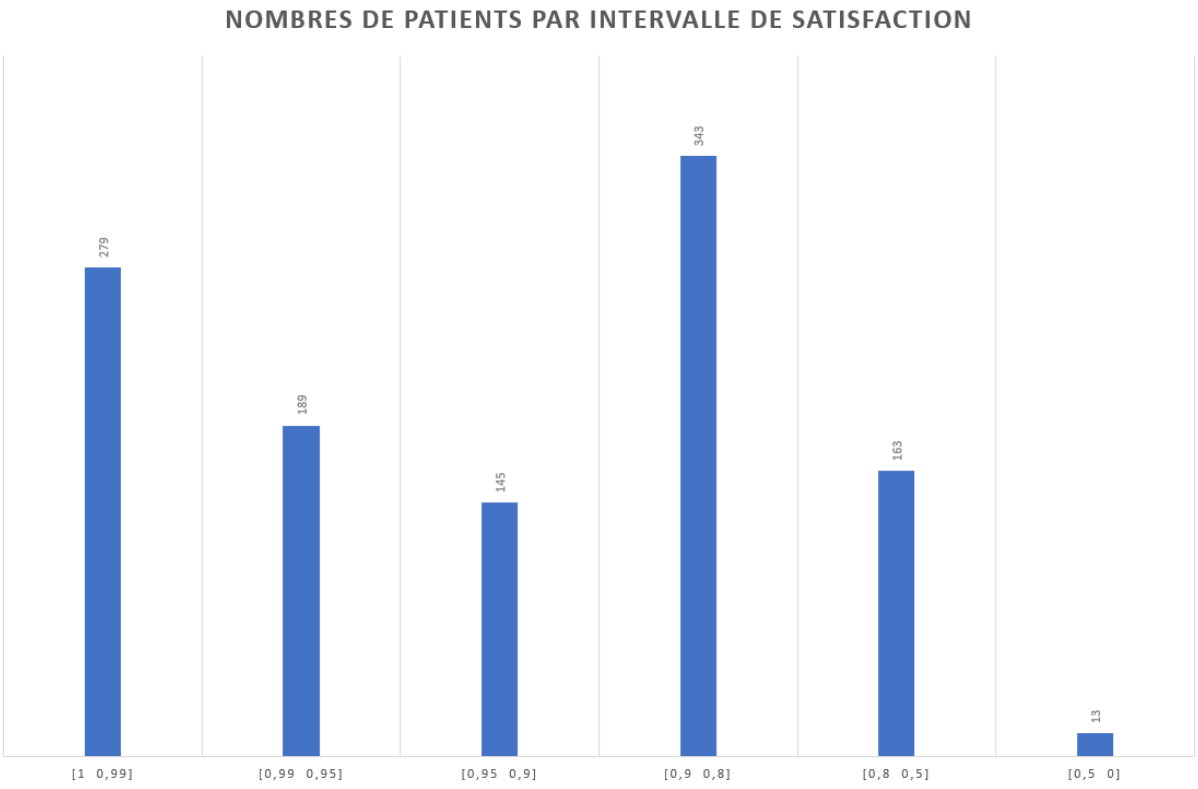

FIGURE 15 - SATISFACTION MOYENNE INITIALE

Cet histogramme représente le nombre de patients par intervalle de satisfaction sur un an. C'est à dire la répartition des curistes en fonction de leur succès à effectuer la totalité de leurs traitements. Nous remarquons que peu de curiste ne parviennent pas à réaliser moins de 50% de leur cure et nous obtenons une efficacité moyenne de 89,7 %. Afin d'améliorer cette efficacité la société DarkForce se penche sur plusieurs options.

Perspectives d'évolution

Afin de diminuer le temps d'attente des patients ainsi que l'efficacité de la cure nous décidons d'utiliser la zone inutilisée à côté des bains à jets anciens en tant qu'autre atelier de bain. Cette décision est prise car d'après l'étude des graphiques précédents les bains sont les ateliers présentant le plus grand taux d'attente des patients. Ceci devrait aussi améliorer l'efficacité de l'institut car si les curistes attendent moins la probabilité qu'ils finissent leur cure est plus grande.

Attente des patients

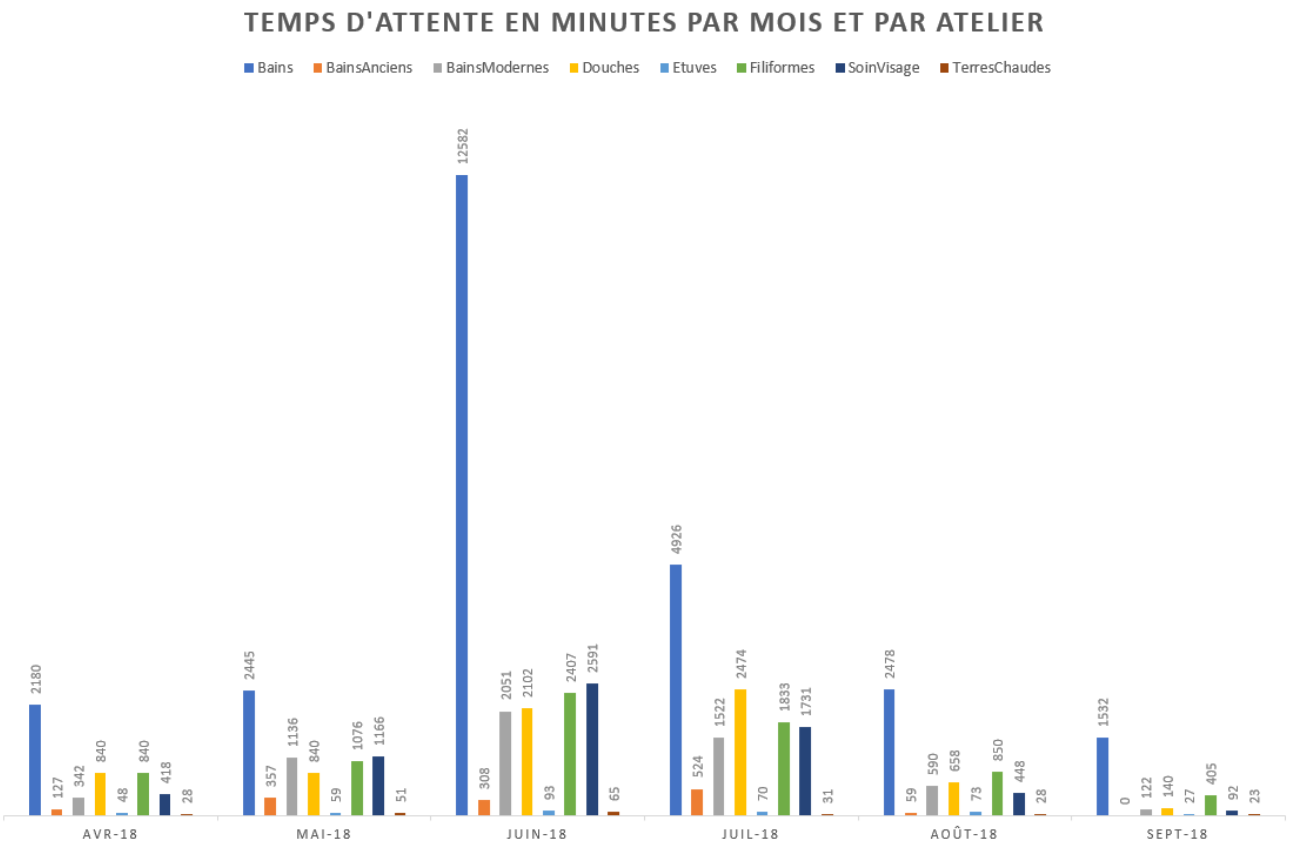

FIGURE 16 - TEMPS D'ATTENTE MOYEN APRES MODIFICATION

Cet histogramme présente le temps d'attente des patients par mois et par ateliers sur un an avec l'ajout de l'atelier "bains" identique aux bains à jets anciens dans la zone inutilisées. Nous remarquons que le temps d'attente diminue énormément suite à cette décision, il est divisé par deux en juin et par 4 en juillet. En moyenne un curiste patiente 10 minutes par mois, le temps d'attente a donc été divisé par deux.

Efficacité de l'institut

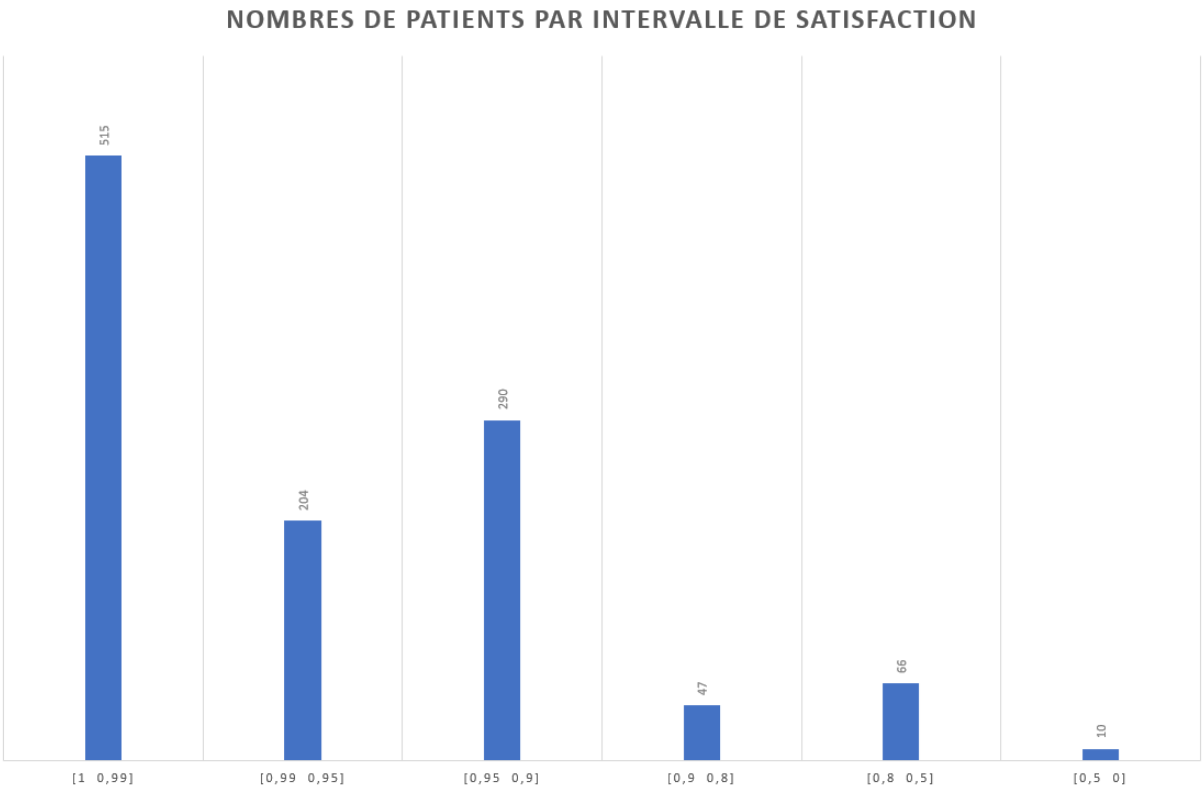

FIGURE 17 - EFFICACITE DE L'INSTITUT APRES MODIFICATION

Cet histogramme présente le nombre de patients par intervalle de satisfaction (taux de réussite d'un patient à effectuer toute sa cure) sur un an. Nous obtenons un bien meilleur résultat que précédemment avec une moyenne de satisfaction de 94.6% et très peu de curiste ayant effectués moins de 80% de leur cure.

Autres pistes d'amélioration

Une autre solution pour améliorer l'efficacité du centre serait d'améliorer la maintenance des ateliers, notamment ceux des étuves et des jets filiformes qui tombent souvent en panne. Ceci permettrait aux curistes d'améliorer leur chance d'effectuer leurs traitements.

De plus, les ateliers à rendez-vous ne souffrent pas de temps d'attente car les patients arrivent à horaires fixes et possèdent des files d'attentes organisées. Il serait raisonnable de faire évoluer les ateliers sans rendez-vous à des ateliers avec, ce qui diminuerait le temps d'attente mais risquerait aussi de diminuer la capacité d'accueil de l'institut.